PER ESPERTI E PRINCIPIANTI

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • APRILE 2005 • ANNO IX, N.4 (90)

ioProgrammo Anno IX - N°4 (90)

PAROLA DI JAVA!

CREA APPLICAZIONI CHE RICONOSCONO LA VOCE ED ESEGUONO I TUOI ORDINI

- Integra un motore di riconoscimento vocale nel tuo codice
- Dai in pasto un dizionario al tuo sofware
- Inizia subito usando gli esempi

AL VIA IL CORSO SYMBIAN

I primi passi per imparare a programmare i cellulari

VB .NET E LA GRAFICA

CAMBIARE MENU

Operare nel cuore delle interfacce per renderle più accattivanti

DATABASE

TUTTI I SEGRETI DI ADO.NET

Programmi veloci come fulmini, sfruttando al massimo le tecnologie di Microsoft

C# INCONTRA MYSQL

Scopri le transazioni: per ottimizzare l'accesso concorrente ai database

IOPROGRAMMO WEB

RETE BLINDATA CON STRUTS!!!

Come usare il framework Java che rende inattacabili i tuoi siti web

TECNICA

APPLICAZIONI CHE FANNO PER TRE!!

Usare i Java thread per fare più cose insieme senza perdere velocità

FILTRI BAYESIANI

Ecco come fanno gli esperti a non ricevere la posta spazzatura

PHP MULTILINGUA

Pronti per i nuovi mercati. Scrivi applicazioni Web che si adattano al paese di chi li visita!!!

HACKING VISUAL BASIC

Incredibile! ti sveliamo le funzioni non documentate della shell

.NET

MS DATABASE APPLICATION BLOCK

Programma velocemente utilizzando i blocchi di codice elaborati da Microsoft

WINDOWS UPDATE AGENT

Scopri Il nuovo SDK per avere il controllo completo sugli aggiornamenti di sistema

CONTINUANO I CORSI
- VISUAL BASIC.NET 2003
- MACROMEDIA FLASH
- ASP.NET

9 771128 594009 00090

EDIZIONI MASTER
www.edmaster.it

Riconoscimento vocale in Java • PHP multilingua • Visual Basic e le funzioni non documentate

KERNEL AI RAGGI X: ALL'INTERNO DEL SISTEMA PER CAPIRNE I SEGRETI

Anno IX - N.ro 4 (90) - Aprile 2005 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Raffaele del Monaco, Fabio Farnesi
Collaboratori: M. Autiero, M. Bigatti, D. Bochicchio, L. Buono, F. Grimaldi, D. Fadda, F. C. Ferracchiati, A. Marroccelli, S. Meschini, V. Muraglia, G. Natili, F. Paparoni, P. Perrotta, M. Poponi, F. Smelzo, L. Spuntoni, A. Trapani, I. Venuti, F. Vaccaro.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €52,90 sconto 30% sul prezzo di copertina di €75,90 ioProgrammo con Libro (11 numeri + libro) €76,00 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/04/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Marzo 2005

Edizioni Master edita: Computer Bild, Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

Piccoli particolari

È incredibile la quantità di email che giungono giornalmente in redazione da parte di persone che si avvicinano per la prima volta alla programmazione. Qualche santone delle statistiche aveva tuonato qualche tempo fa: "non si fa programmazione in Italia". Forse, avrà avuto il suo bravo elenco di numeri da analizzare per affermare qualcosa del genere. Invece noi che in mezzo alla programmazione ci siamo tutti i giorni, avvertiamo un entusiasmo tangibile, presente, forte, in questo segmento che così tanto ci attrae. Mi sono sempre chiesto cosa potesse stimolarmi a tuffarmi in un universo complesso quale è quello fatto di codice, algoritmi, numeri, ragionamento. Mi sono risposto spesso e volentieri, forse peccando di presunzione, che quello che mi rende totalmente avvolto al mondo dello sviluppo è la possibilità di creare qualcosa. Un prodotto, un software, nasce da un'idea, viene sviluppato secondo una forma che per certi versi è simile all'arte, viene affinato e coccolato come il più sentito dei capolavori. Con questo non voglio assimilare il nostro lavoro alla genialità immortale degli artisti, non arrivo a tali vette di presunzione, ma è anche vero che sentire di essere capace di produrre qualcosa che aiuta a migliorare in un qualche modo la qualità della vita di questo nostro mondo, è proprio quello che mi tiene avvinto alla complessità della sfida con la programmazione. È che il software aiuti a migliorare la qualità della vita è innegabile, guardatevi intorno, in tutto o quasi tutto quello che vi circonda c'è dentro almeno una riga di codice, e se non c'è, sicuramente qualche riga di codice sarà stata spesa per produrre l'oggetto in questione. Perciò il mio invito è quello di "darci dentro". Nuovi programmatori, non fermatevi davanti alle difficoltà, qualcosa potrebbe apparirvi complesso, non tutto vi sarà chiaro, ma è proprio il fascino di questa sfida, la volontà di conoscere al fine di creare che ci consente di dire con orgoglio di far parte del mondo dello sviluppo!

Fabio Farnesi

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **gatto** *Password:* **matita**

PAROLA DI JAVA!

Creare applicazioni che riconoscono la voce e rispondono la cosa giusta

- **Il motore di riconoscimento vocale Sphinx4**
- **Definire le grammatiche**
- **Riconoscere i comandi**

pag. 16

HACKING VISUAL BASIC

Ti sveliamo le funzioni non documentate della shell pag. 66

http://forum.ioprogrammo.it

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

RUBRICHE

ioProgrammo cerca articolisti freelance competenti nei seguenti argomenti:

Javascript, Python, Perl, ASP.NET, PHP, Flash, Security

Inviare curriculum dettagliato a ioProgrammo@edmaster.it

I contenuti del CD-Rom

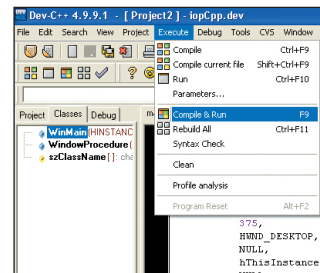
Dev C++ 5 Beta 9
Un editor C++ a basso costo

Directory: /Devc

MySQL 4.1.9
Il server di database OpenSource più diffuso al mondo
Directory /Mysql

MySQL Administrator
Il front-end per amministrare mysql in modo grafico
Directory: /mysql-administrator

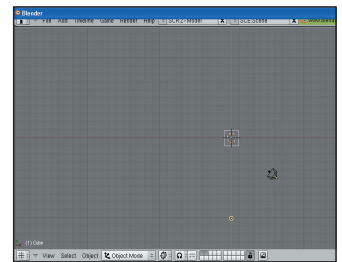
MyODBC
Il connector universale per MySQL
Directory: /MyODBC

MySQL Query Browser
L'editor SQL per MySQL
Directory: /mysql-query-browser

Sqlite 2.8.15
Il nuovo database Bundled con PHP5
Directory: /Sqlite

HSqldb 1.7.3
Un database piuttosto potente
Directory /hsqldb

SharpDevelop 1.0.3.1761
L'alternativa a Visual Studio a basso costo
Directory: /Sharpdevelop

Aqua Data Studio 4.0
Il query builder leggero ed efficiente, quasi un indispensabile
Directory /Acquadatastudio

Blender 2.36
Un modeller per oggetti 3D. Molto simile a 3D Studio

Directory: /Blender3D

Apache 1.3.33/2.0.52
Directory: /Apache/

Tomcat 5.5.4
Il servlet container per Java e JSP
Directory: /tomcat

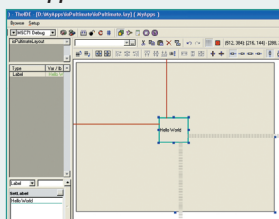
PHP 5.0.3
Il linguaggio di scripting per il web
Directory: /PHP

Python 2.4
Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà
Directory: /Python

Hibernate 3.0
Il framework per gestire la persistenza dei dati
Directory /Hibernate

Il software del mese

Ultimate++
Un fantastico ide "quasi RAD" per le applicazioni C++

Si tratta di un ambiente di sviluppo per applicazioni C++, nella versione allegata a ioProgrammo lo trovate abbinato al compilatore Mingw, nonostante questo può essere utilizzato anche con altri compilatori.
L'ambiente offre tutte le caratteristiche classiche di un ambiente professionale, parliamo di code complexion, debugging, sintax highlighting.
La caratteristica più interessante dell'IDE è che è dotato di un minimo di funzionalità RAD il che dato i bassi costi del pacchetto lo rende particolarmente attraente agli occhi degli sviluppatori.

Directory: /Ultimate [pag.116]

BORLAND JBUILDER 2005 FOUNDATION EDITION
L'eccezionale ambiente di Borland per lo sviluppo Java

Jbuilder è l'ambiente proposto da Borland per lo sviluppo di applicazioni Java. Come è tradizione di Borland si tratta di un ambiente incredibilmente curato nei particolari. Dotato di tutte le caratteristiche essenziali di un ottimo editor Java, la versione Foundation ha dalla sua parte il costo praticamente nullo, dovuto alla sua licenza, l'uso della foundation è infatti gratuito per usi personali. Certo ha delle limitazioni rispetto alla versione enterprise, ad esempio non è possibile gestire i progetti UML, ma si tratta di limitazioni che comunque influiscono solo sulla produzione di progetti di largo respiro. Insomma per un programmatore Java che si accinge a iniziare e che non ha necessità di sviluppare progetti Enterprise, la versione Foundation è assolutamente da provare.

Nello spirito tipico degli ambienti prodotti da Borland, anche JBuilder é un tool che segue la filosofia che le applicazioni vanno disegnate e che la produttività di un programmatore dipende anche da quanto tempo riesce a risparmiare nel disegno dell'Interfaccia, pertanto JBuilder é dotato di un Visual Designer che consente di disegnare le form per trascinamento dei componenti. Solo vantaggi per JBuilder, unico punto a sfavore é che richiede un sistema sufficientemente dotato per potere essere utilizzato con profitto

Directory /JbuilderFoundation [pag.115]

Lazarus 0.9.4
Un ambiente RAD piu copilatore Freeware per object Pascal

Volevate imparare a programmare in Delphi ma non avevate la possibilità di comperare il costoso ambiente di casa Borland? Lazarus è ciò che fa per voi. Si tratta di un clone Freeware del noto Delphi. La somiglianza è incredibile! Il modo di procedere altrettanto! Lazarus è un RAD funziona con la stessa logica di Delphi, supporta la VCL, è dotato di tutti i componenti classici di Delphi, è sufficientemente veloce e affidabile. Certo, non è il Borland Delphi 2005 con tutte le sue infinite possibilità, ma rappresenta un'ottima base per chi vuole sviluppare applicazioni Standalone utilizzando la produttività classica di un ambiente RAD

Directory /Lazarus [pag.115]

I Kit del programmatore

Gli strumenti essenziali sia per il professionista sia per chi vuole cominciare

PHP 5.0.3
L'ultima versione del linguaggio

DEV-PHP 2.0.9
Uno degli editor per PHP più amati dai programmatori

DEV C++ 5 BETA 9
Lo standard di cui non puoi fare a meno

ULTIMATE C++
Nuovissimo ed addirittura RAD

SHARP DEVELOP 1.0.3.1761
L'alternativa a Visual Studio a basso costo. Programma in .NET in modo rapido e visuale

LAZARUS 0.9.4
Un ambiente RAD, più un copilatore Freeware per object Pascal. Quasi come Delphi ma gratis!

J2SE 1.5.0
Indispensabile per programmare in JAVA

ECLIPSE 3.0.1
La piattaforma universale

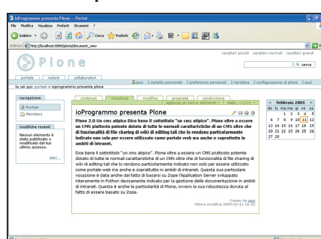
Dev-PHP 2.0.9
Ottimo editor PHP OpenSource

Se state iniziando a sviluppare in PHP avrete bisogno di un editor. Scrivere codice con il notepad può essere un esercizio divertente, ma quando iniziate a scrivere script leggermente più complessi si impone la scelta di passare a un editor più completo. DEV-PHP non solo è completo ma anche molto potente.
Dotato di code completion, sintax highlighting, funzionalità di ricerca avanzate ed una serie di tool piuttosto interessanti rappresenta una grande scelta per programmare in PHP. Inoltre è un editor straordinariamente leggero, oltre che gratuito ed OpenSource. In molti lo troveranno molto comodo.
Da non perdere!

Directory: /DevPHP [pag.116]

Spe 0.7.0
Un editor evoluto per Python
Directory: /Spe

Eclipse 3.0.1
La piattaforma universale multifunzione
Directory: /Eclipse

Plone 2.0
Un cms atipico

Directory /Plone

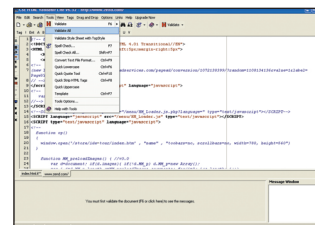
Irrlicht 0.7
Sviluppare Giochi 3D potenti in modo semplice
Directory: /irrlicht

Smarty 2.6.7
Il template engine per PHP
Directory /Smarty

Sqlite .NET
Un connector .NET per questo interessante database
Directory: /Sqlite

Menalto Gallery 2.0
Una completa gallery fotografica pronta per essere usata sul Web
Directory: /gallery

TikiWiki
Un Wiki piuttosto interessante
Directory /tikiwiki

Cse HTML Validator Lite V6.52
Un validatore per le vostre pagine html

Directory /csevalidator

MySQL Connector
Per utilizzare MySQL direttamente da applicazioni .NET, Java, Python
Directory: /Connector

Sphinx4
Il motore di riconoscimento vocale
Directory: /Sphinx4

Lua
Per dotare i propri programmi di un ambiente di scripting
Directory: /Lua

PHP-GTK
Creare GUI con PHP
Directory: /PHP-GTK

Tortoise CVS
Ottimo tool per il controllo della revisione
Directory: /tortoisecvs

Phpmq
Integra JMS in applicazioni PHP
mantaray_1.2.1_bin.zip

Prova subito

DAMN SMALL LINUX
Linux in 50 MB: una delle distribuzioni più piccole al mondo

Masterizza la traccia ISO, inserisci il CD e goditi l'ebrezza del pinguino.

File su CD: /damnsmall [pag.116]

News

IN ARRIVO IL CELLULARE CON LINUX

Direttamente dalla Germania e precisamente da Road GMBH è in arrivo un prodotto molto simile al conosciutissimo Nokia Communicator ma basato su Linux e addirittura su un sufficientemente recente kernel 2.6. Il prodotto integrerebbe GSM, GPRS, IrDA, Bluetooth, WLAN, tutte le applicazioni tipiche come un browser per Internet, un client Email un viewer per MS Word e per Excel, il classico PIM e tutte le applicazioni più comunemente usate. L'interfaccia grafica sarebbe però basata su Qtopia mentre il cuore sarebbe il noto Linux con Kernel 2.6. Il prodotto porta la sigla di S101 e sono previste versioni basate anche su Symbian e su Windows CE.

È ACCORDO FRA IBM E ZEND

Le strade dell'informatica sono infinite! Cosa hanno in comune Big Blue, colosso dell'informatica aziendale e Zend leader degli strumenti di sviluppo per PHP? Fino a ieri niente. Da oggi hanno in comune un accordo per sviluppare insieme strumenti per aiutare gli sviluppatori a scrivere applicazioni basate sul popolare linguaggio PHP. La notizia è di quelle che scotta. Il solo PHP rappresenta il maggior concorrente di Microsoft per quanto riguarda lo sviluppo di applicazioni Web, l'accordo con Zend rappresenta un muro difficile da superare per il colosso di Redmond. Inoltre la partnership di IBM con Zend garantirà a PHP di occupare fasce di mercato che fino ad ora aveva soltanto sfiorato, vedi il mercato Business /Aziendale.

CIA LA PRIMA SOLUZIONE ITALIANA OPENSOURCE DI IDENTITY MANAGEMENT

Inutile ricordare quanto e come Internet sia cambiata dalla sua nascita e continui ad evolvere con eccezionale rapidità. Altrettanto vero è che cresce l'offerta di prodotti legati al mondo Internet, siano essi servizi o prodotti materiali. Allo stesso modo crescono le soluzioni di intranet aziendale basate sulle stesse identiche metodologie applicate al mondo Internet. Se tutto questo non può che aiutarci a migliorare la qualità della vita, d'altra parte è inevitabile che questo sviluppo stimoli la nascita di ulteriori e fin qui sconosciute problematiche. La gestione delle identità delle persone è una di queste. Come gestire a livello aziendale la registrazione degli utenti? Come dialogare con la crescente necessità di privacy? Come assicurarsi della corretta identità degli utenti? E ancora come sviluppare applicazioni che supportino gli utenti nella gestione delle loro identità?
Ad esempio è opportuno sviluppare applicazioni che consentano agli utenti di cambiare le loro password, i loro dati personali senza per questo dovere passare attraverso un helpdesk. Tutte queste problematiche sono generalmente assistite da software di Identity Management. Sys-Net è una delle aziende Italiane che si è maggiormente distinta in questo settore e annovera clienti di grande livello come Banca Intesa, Gruppo San Paolo e molti altri ancora. Colpisce la volontà di puntare sull'Open Source come scelta strategica anche per lo sviluppo di soluzioni così delicate.
Eppure Barbara Dell'Era, Direttore Commerciale della Sys-Net si dichiara convinta che la direzione intrapresa porterà da un lato un enorme sviluppo tecnologico del progetto, dall'altra una crescente possibilità di customizzazione della soluzione. Noi, ovviamente siamo d'accordo con lei.

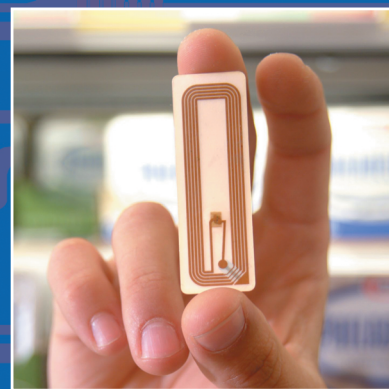
BEN GOODGER ABBANDONA FIREFOX E PASSA A GOOGLE

"È stata un'esperienza enteressante, emozionante ed educativa... tuttavia quando un progetto durato molto tempo raggiunge un suo traguardo essenziale è normale che la gente che vi ha lavorato prenda una pausa di riflessione , si guardi intorno e decida cosa vuole fare" . Questo si legge sul Blog di Ben Goodger, direttore esecutivo del gruppo che ha portato alla nascita di FireFox il più temibile rivale di Internet Explorer *http://weblogs .mozillazine.org/ben/archives/007366.html* . Segue l'annuncio di essere stato ingaggiato dal noto Google. Al momento Goodger è stato evasivo sul ruolo che avrà all'interno di Google, viceversa è emerso chiaramente che nonostante il suo impegno in Google continuerà a lavorare ancora nel gruppo che ha portato all'avvento di Firefox 1.0 almeno fino al rilascio della versione 2.0 passando attraverso la 1.5.
Certo è che la notizia è di quelle che lasciano senza fiato, non si sa cosa potrebbe partorire dai genietti che hanno fatto nascere il più importante motore di ricerca al mondo e dalla fantasia del capo progetto di FireFox. A nostro avviso possono nascere solo guai per Microsoft, ma questa è solo una nostra sensazione.
Staremo a vedere.

NO AI BREVETTI SUL SOFTWARE

Da qualche tempo ormai ci portiamo dietro la diatriba che contrappone l'Unione Europea alle associazione per la promozione del software libero in materia di brevettabilità del software.
Per il momento sembra avere avuto ancora una volta ragione il movimento che si oppone al testo di legge sulla brevettabilità.
Di fatto il parlamento europeo avrebbe azzerato, grazie alle numerose mozioni arrivate un po da tutti i paesi europei, l'attuale proposta.
Si dovrà dunque ricominciare da zero, tenendo questa volta conto delle obiezioni mosse da chi ritiene che il software sia un bene troppo prezioso e troppo utile all'umanità perché lo sviluppo possa essere fermato da lacci e lacciuoli burocratici. Il sito di riferimento per comprendere le ragioni di chi si oppone alla brevettabilità del software è *http://www.nosoftwarepatents.com/it/m/intro/index.html*

RFID FUORI DALLA SCUOLA ELEMENTARE

Una scuola Elementare in California aveva avviato un esperimento per dotare gli alunni di un Badge connesso a un Rfid. Lo scopo era ovviamente quello di tenere sotto controllo gli spostamenti degli alunni e di automatizzare alcuni processi interni. Immediatamente dopo l'attuazione del provvedimento era montata la protesta dei genitori che lamentavano una violazione della privacy nei confronti dei loro ragazzi. Il preside era stato costretto ad eliminare i sensori che leggevano le informazioni provenienti dagli RFID dalle porte dei bagni dove erano stati montati per mantenerli attivi solo nelle porte delle aule. Poco dopo l'esperimento è stato sospeso ed è stato rescisso il contratto all'azienda che aveva ottenuto l'appalto del progetto adducendo la motivazione di "scontento" nel come lo sviluppo era stato portato avanti. Infine la colpa di tutto quello che non funziona è sempre dei programmatori!

MOTOROLA VERSO IL VOICE OVER IP

Alcuni di voi forse conosceranno Skype. Non ancora diffusissima in Italia, conta però nel mondo circa 25 milioni di utenti registrati. Skype è una giovane società che fornisce servizi a basso costo di telefonia su Internet basati ovviamente sulla tecnologia Voice Over IP.

In Italia ci sono stati fino ad ora diversi tentativi di avviare in modo massiccio il VoIP ma nessuno di questi ha realmente fino ad ora riscosso un enorme successo. Se le tendenze sono quelle di oltreoceano è però molto probabile che anche da noi in tempi non eccessivamente lunghi faranno la loro comparsa servizi affidabili che consentano di telefonare utilizzando Internet e ad un costo sicuramente inferiore a quelli attuali. Nel frattempo ancora una volta oltreoceano desta un certo interesse l'annuncio che Motorola, colosso della telefonia mobile produrrà telefonini Skype Enabled. Al momento non c'è una vera precisa offerta commerciale su come questo possa influire sulle abitudini dell'utente finale, tuttavia c'e' un accordo per cui Motorola produrrà, cuffie e microfoni certificati per essere usati con tecnologia VoIP e in comarketing con Skype. E' curioso mostrare come la tecnologia stia evolvendo in questa direzione. D'altra parte quello che vediamo oggi è solo un germe delle tecnologia che saranno da qui a quattro o cinque anni, d'altra parte il primo televisore misurava appena nove pollici no? Sicuramente in breve tempo le nostre abitudini di vita saranno molto cambiate in relazione a quelle che al momento sembrano soltanto fantasiosi rumors.

NASCE GRUSP IL "GRUPPO UTENTI E SVILUPPATORI PHP"

Nato come associazione senza fini di lucro e con il chiaro intento di diffondere il PHP all'Italia e all'estero, il "Gruppo Utenti e Sviluppatori PHP" ha il suo baricentro nel sito *http://www.grusp.it*.
Non si tratta del solito sito, ma di una community organizzata sotto la forma di Wiki, dove ognuno può contribuire con articoli, documentazione, link trucchi è consigli. È interessante sia la forma associativa sotto cui il GRUSP nasce, ovvero un'associazione senza fini di lucro legalmente riconosciuta, il che prelude alla volontà di dare continuità all'iniziativa, sia l'idea di basare la community su un Wiki piuttosto che sul classico blog o su un CMS chiuso. Il wiki consente di dare continuità anche agli articoli proposti grazie al contributo libero che ciascuno può dare nella proposizione di questa o quella precisazione. In bocca al lupo Grusp!

INBox
L'esperto risponde...

Quale linguaggio?

Sviluppo applicazioni ma ora sono arrivato ad un punto molto importante: conosco e programmo in quasi tutti i linguaggi esistenti ma vorrei specializzarmi su una piattaforma in particolare. Avevo pensato a .net affinando l'accoppiata C# e ASP.NET, poi ho pensato alla possibilità di sviluppo multipiattaforma windows-linux utilizzando java e jsp, infine c'è una terza strada rappresentata da PHP sempre potente e di facile comprensione. Vorrei sapere da voi quale è la direzione del mercato e verso che linguaggio di programmazione conviene orientarsi. Ringraziando anticipatamente.

Giacomo

Non è facile rispondere. Tutti i linguaggi hanno vantaggi e svantaggi e ciascuno viene utilizzato per le proprie pecurialità. Le ultime classifiche per quanto riguarda la diffusione dei linguaggi riportano la seguente statistica:

1	*C*	*+1.63%*
2	*Java*	*-4.22%*
3	*C++*	*-4.37%*
4	*PHP*	*+3.02%*
5	*Perl*	*-0.61%*
6	*Visual Basic*	*-2.04%*
7	*SQL*	*-0.14%*
8	*Python*	*+1.50%*
9	*C#*	*+0.19%*
10	*Delphi/Kylix*	*+1.17%*

Fonte tiobe.com.

Nel tempo medio/lungo la massima crescita sembrerebbe essere attribuibile a Python, immediatamente seguito da PHP. Tuttavia non si può dire che questi due siano i linguaggi su cui puntare, molto dipende sulla tipologia di applicazione da realizzare. Per quanto riguarda il Web, PHP rappresenta senza dubbio un'ottima scelta. È anche vero che se devi sviluppare applicazioni molto legate agli ambienti Microsoft è opportuno puntare decisamente su Visual Basic.NET o C#. Per le applicazioni Web che necessitino di particolare sicurezza JSP è sicuramente un'ottima soluzione.
Python è una scelta interessante per la creazione di script di sistema o di interfacce di frontend verso applicativi formati da moduli separati. Come vedi non è possibile dare una risposta precisa. È importante capire esattamente quali sono le richieste che devi soddisfare, e poi scegliere il linguaggio adeguato che rende meno problematico lo sviluppo dell'applicazione.

C# o VB.NET: questo è il dilemma

Cara ioProgrammo, mi sono da poco avvicinato al mondo della programmazione e vorrei cimentarmi con la piattaforma .NET. Sono un po' in dubbio sul linguaggio da scegliere: C# o Visual Basic? Voi cosa ne dite? Quali sono le differenze fra i due e cosa consigliereste a chi, come me, deve affrontare la scelta? Un grazie anticipato e mille complimenti per l'ottimo lavoro!

Mario

I due linguaggi che hai menzionato hanno molte somiglianze. Per capirne le differenze, partiamo dalla genesi: VB.NET è stato progettato per andare incontro alle esigenza degli sviluppatori Visual Basic, ampliando enormemente le possibilità offerte dalla vecchia piattaforma VB 6. Visual C# è stato invece progettato con una sintassi e filosofia della piattaforma Java (Questa affermazione non la sentirete mai dalla bocca di un progettista Microsoft… ma siamo molto vicini alla realtà). Inoltre, tutto il framework .NET ricalca l'impostazione che Sun ha dato al suo gioiellino, in primis con l'utilizzo della Virtual Machine. Quindi, per tornare al consiglio che chiedevi, direi che se cominci da zero puoi senz'altro adottare C# come linguaggio: forse richiederà uno sforzo iniziale leggermente più grande, ma verrai ripagato ampiamente dalla migliore aderenza della sintassi alla filosofia .NET. Scegliendo Visual Basic .NET avrai pure sempre un ottimo linguaggio e potrai ottenere gli stessi risultati di C#, il pegno da pagare è però una certa forzatura nel piegare la sintassi Visual Basic alla semantica richiesta da .NET che è strettamente orientata agli oggetti. Insomma, se vuoi partire subito a razzo, prova con VB.NET, se invece puoi dedicare un po' di tempo e un po' di studio, ti consiglio vivamente di "attaccare" C#, uno dei migliori linguaggi in circolazione.

PHP & XML

Buongiorno amici di ioProgrammo. Sto cercando di capire come parserizzare un file XML con PHP, e devo dire di avere piu o meno ben chiaro tutto. Quello che non riesco a capire è come funziona XPath. Cioè mi pare di avere capito che posso fare delle interrogazioni su un file XML come se le facessi in SQL su un database, quello che non ho capito è come farlo in PHP e come funziona la sintassi di XPath.

Romeo

Buongiorno Romeo. Direi che il tuo intuito non ha sbagliato. XQuery è un linguaggio di interrogazione per documenti XML. Si tratta di una tecnica abbastanza potente che consente di recuperare un insieme di dati da un file XML senza doverlo parserizzare interamente e soprattutto recuperando stret-

tamente i nodi interessati dalla query. Viceversa utilizzando le normali interrogazioni non sempre é possibile essere così precisi nel prelevare le informazioni che ci servono da un file XML senza dover ricorrere costosi costrutti condizionali. Un esempio sarà chiarificatore:

```
<?php
$xmldocument = new DOMDocument();
$xmldocument->load(
    'http://www.theregister.co.uk/excerpts.rss');
$xp = new DOMXPath($xmldocument);
$nodelist = $xp->query(
                    "//rss/channel/item/title");
foreach ($nodelist as $node) {
            print $node->textContent.'<br>';
}
?>
```

Questo spezzone di codice recupera tutti i nodi *title* da un documento XML. Possiamo utilizzare una tecnica ancora più potente, utilizzando una query complessa.

```
<?php
$xmldocument = new DOMDocument();
$xmldocument->load(
  'http://www.theregister.co.uk/excerpts.rss');
$xp = new DOMXPath($xmldocument);
$nodelist =
$xp->query("//rss/channel/
                item[substring(pubDate,9,3)
                            ='Jan']/title");
foreach ($nodelist as $node)
{
  print $node->textContent.'<br>';
}
?>
```

Recupera tutti i noti title tali che il campo pubDate di idem contenga la stringa 'Jan' La sintassi non è immediatamente intuitiva. Un buon libro che spiega questi aspetti è quello pubblicato da Gianfranco Forlino per la collana "i libri di ioProgrammo" e dal titolo: "XML Guida alla Programmazione".

Controllare la registrazione di un dominio

Salve cari Guru della programmazione. Chiedo a voi lumi su una funzione che non riesco a realizzare. Vorrei creare un'applicazione che consenta ad una persona di sapere se un dominio internet è libero oppure no. Ci sono parecchi siti su internet che svolgono questa funzione, ma non ho mai capito come fanno. Mi date una dritta?

Giuseppe

Ciao Giuseppe. Parliamo della situazione italiana. In Italia l'ente che si occupa di mantenere il registro dei domini attribuiti è il NIC *http://www.nic.it*. Ovviamente non si occupa di mantenere semplicemente il registro, ma gestisce le deleghe e l'approvazione delle registrazioni, le cancellazioni e tutto quello che ha a che fare con i domini internet. Delega ai singoli mantainer, quelli a cui ci si rivolge di solito per la registrazione, il compito di fare da intermediari fra l'utente finale e il nic. Il concetto di delega è più ampio, ma direi che per la nostra spiegazione possiamo assumere questo come valido.
Il registro mantenuto dal Nic può essere interrogato mediante protocollo "whois". "Whois" è un servizio che generalmente gira sulla porta 43 del sistema che lo mantiente. Esiste ovviamente il dominio "whois.nic.it" che mantiene un servizio di whois sulla porta 43 di un server che interroga il registro dei domini registrati.
Interrogando un servizio whois si ottiene come risultato un testo che contiene lo stato del dominio. Per cui, riepilogando, per controllare se un dominio è registrato oppure no, è opportuno controllare il registro dei domini tramite il protocollo whois. Perciò sarà necessario aprire un socket verso la porta 43 del server che mantiene il servizio whois, e poi parserizzare il file di testo risultante alla ricerca della stringa che testimonia se il dominio è stato registrato oppure no. In PHP ad esempio:

```
$domain = "www.pippopappo.it";
$fp = fsockopen("whois.nic.it", 43,
                            $errno, $errstr, 30);
fputs($fp, "$domain\r\n");
while (!feof($fp)) {
   $buf = fgets($fp,128);
   if (ereg("No entries found ", $buf))
  {
     $result = "Non registrato";}
}
```

Goto... in Java!!!

Gentile Redazione, sono un "vecchio" programmatore Visual Basic che, tra sforzi e invettive, si sta convertendo a Java. Con mio sommo dispiacere, ho notato che in questo linguaggio (per altro veramente eccezionale) manca l'istruzione GOTO... mi sbaglio? Vi prego, ditemi di si!

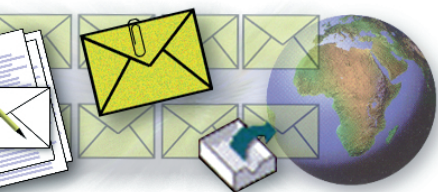
Riccardo

Gentile Riccardo, devo dire che non ti sbagli! In Java, il commando GOTO non è presente e, fossi in te, non me ne rammaricherei più di tanto. Lo sforzo di costruire codice che non faccia uso di salti condizionati non potrà che giovare al tuo stile di programmazione.
Comunque, se proprio non puoi farne a meno, puoi ricorrere ai break con etichetta che consentono di sporcare il codice a sufficienza!
Di seguito trova un esempio

```
class JavaGoto {
 public static void main(String args[]) {
   int max = 20;
   int limite = 10;
   int i = 0;
   out: {
      for( int riga=0; riga< max; riga++ )
      {
         for( int col=0; col< max; col++ )
         if( riga== limite) break out;
         j += 1;
      }
   }
   System.out.println(i);
// il ciclo si ferma con i pari a 5
 }
}
```

In questo caso il break riesce a rompere ben due cicli innestati... non so se fa al caso tuo, ma è quanto di più vicino al Goto esista in Java!

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Ariberto, 24 - 20123 Milano

NEWSGROUP
Le informazioni nella Rete

Direttamente dai forum di ioProgrammo le discussioni più "Hot" del momento

Come ottenere il percorso .NET Framework

Avrei bisogno di un metodo veloce per conoscere in modo programmatico la posizione del .NET framework nel computer che esegue un'applicazione, come posso fare?

Risponde Salvatore Meschini
http://forum.ioprogrammo.net/thread.php?threadid=4726&boardid=36

Il sorgente va salvato in un file (es. *getpath.cs*). Si digiti: *csc /target:winexe getpath.cs* In alternativa scaricate il programmino Snippet Compiler da *http://www.sliver.com*. Il pezzo di codice che risolve il problema è il seguente:

```
using System;
using System.IO;
using System.Windows.Forms;
public class WindowsForm :
                    System.Windows.Forms.Form
{
   public static void Main() {
      Application.Run(new WindowsForm());
   }
   private string CLRPath() {
      Object CLR = new Object();
      return Path.GetDirectoryName(
           CLR.GetType().Assembly.Location);}
    public WindowsForm() {
       MessageBox.Show(CLRPath());
    }
}
```

Elencare le unità di rete con WMI

Talvolta può essere necessario disporre dell'elenco delle unità di rete configurate sul proprio PC, determinarne lo stato della connessione, se questa verrà ripristinata al riavvio del sistema o conoscere il percorso remoto del disco mappato come unità. Per ottenere in maniera semplice queste ed altre informazioni con Visual Basic, si può ricorrere a WMI

SimoneVB
http://forum.ioprogrammo.net/thread.php?threadid=4703&boardid=36

Il codice proposto da SimoneVB è il seguente:

```
Private Sub Form_Load()
If Not IsWMIInstalled() Then
MsgBox "L'applicazione richiede Microsoft
    Windows Management Instrumentation
                        (WMI).", vbCritical
Exit Sub
End If
Dim oLoc
Dim oServ
Dim oObjectSet
Dim oConn
Dim sMsg As String
Const sWQL As String = "SELECT * FROM
 Win32_NetworkConnection" ' query WQL
Set oLoc = CreateObject("WbemScripting
         .sWbemLocator") ' oggetto locator
Set oServ = oLoc.ConnectServer(
                       ".", "root\cimv2")
Set oObjectSet = oServ.ExecQuery(sWQL)
For Each oConn In oObjectSet
Print
sMsg = "Unità di rete: " & oConn.LocalName
sMsg = sMsg & vbCrLf & "Path remoto: "
                      & oConn.RemotePath
sMsg = sMsg & vbCrLf & "Ripristino
     connessione all'avvio di Windows: "
                      & oConn.Persistent
sMsg = sMsg & vbCrLf & "Tipo di risorsa: "
                    & oConn.ResourceType
sMsg = sMsg & vbCrLf & "Stato: "
                          & oConn.Status
sMsg = sMsg & vbCrLf & "Stato
 connessione: " & oConn.ConnectionState
sMsg = sMsg & vbCrLf & "Commento: "
                         & oConn.Comment
MsgBox sMsg, vbInformation, "Unità di rete"
Next
Set oConn = Nothing
Set oObjectSet = Nothing
Set oServ = Nothing
Set oLoc = Nothing
End Sub
Function IsWMIInstalled() As Boolean
Dim oTemp
On Local Error Resume Next
Set oTemp = CreateObject(
            "WbemScripting.sWbemLocator")
IsWMIInstalled = (Err.Number <> 429)
If Err.Number = 429 Then
Err.Clear
Else
Set oTemp = Nothing
End If
End Function
```

Tomcat con Windows XP

Salve, qualcuno ha provato a far funzionare php con Tomcat su Windows XP? Poi mi sono arreso l'ho configurato con Apache e in 3 secondi ha funzionato. Avrei però bisogno di integrare PHP e TOMCAT. Qualcuno mi può aiutare? Grazie.

Santix
http://forum.ioprogrammo.net/thread.php?threadid=4809&boardid=2

Risponde doc

Non so quale procedura tu abbia usato, io tempo fà ne ho usata una riportata sul Wiki di Jakarta e ha funzionato:

- Devi definire le solite variabili d'ambiente

```
$JAVA_HOME,$TOMCAT_HOME, $PHP_HOME
```

- Per quanto riguarda PHP al configure devi passare i seguenti parametri

```
./configure --with-servlet=$TOMCAT_
           HOME --with-java=$JAVA_HOME
```

- Quindi dopo il make hai due file che ti servono, *sapi/servlet/phpsrvlt.jar* e *libs /libphp4.so*
- copi il file jar dentro *$TOMCAT_ HOME/common/lib*
- copi *$PHP_HOME/sapi/servlet/web .xml* (servlet e servlet-mapping) dentro il file *$TOMCAT_HOME /conf/web .xml*
- ridefinisci la variabile *LD_LIBRARY_ PATH=$PHP_HOME/libs* a livello di sistema .

Poi c'è anche uno script per cambiare la configurazione di PHP, visto che dalla versione 4.x alla 5.x Tomcat ha cambiato la locazione e il nome del jar riguardante le servlet

```
--- configure.org 2004-04-07
                    11:20:24.000000000 +0200
+++ configure 2004-04-07
                    11:22:50.000000000 +0200
if test "$withval" = "yes"; then
SERVLET_CLASSPATH=.
else
+ if test -f $withval/common/lib
                         /servlet-api.jar; then
+ SERVLET_CLASSPATH=$withval
                  /common/lib/servlet-api.jar
+ fi
+
if test -f $withval/lib/servlet.jar; then
SERVLET_CLASSPATH=$withval/lib/servlet.jar
fi
```

Collegare due computer in Rete

Ciao a tutti...vi voglio porre un problema... possiedo due computer, uno equipaggiato con w2k e sp4 e l'altro con XP, e un hub... voglio collegarli e creare una piccola rete ma dopo innumerevoli tentativi ancora non ci riesco... . Qualcuno potrebbe illuminarmi magari spiegandomi i passi necessari?
Grazie e ciao a tutti

AironeAngelo
http://forum.ioprogrammo.net/thread.php?threadid=4875&boardid=31

Risponde jachetto
Devi configurare il TCP/IP sulle due macchine come segue

TCP-IP sulla prima macchina

IP: 192.168.1.2
GateWay: 192.168.1.1
NetMask: 255.255.255.0
DNS: quello che ti dice il tuo provider

TCP-IP sulla seconda macchina
IP: 192.168.1.3
GateWay: 192.168.1.1
NetMask: 255.255.255.0
DNS: Stesso DNS di prima

Poi fai un ping dalla prima alla seconda da una console di MSDOS

```
ping 192.168.1.3
```

se risponde sei a posto, altrimenti qualcosa non va. Controlla che i cavetti siano buoni, oppure controlla la scheda di rete. Ovviamente se devi andare in internet una delle due deve fare da router, oppure devi prendere un router esterno.
Se una delle due deve fare da router, il wizard per la condivisione della connessione farà le impostazioni per te.

Trovare le cinque directory più recenti

Salve a tutti. Vorrei sviluppare un piccolo software per fare una ricerca nel sistema e trovare le 5 directory piu' recenti. Il problema e' che non so proprio da dove iniziare. Io con l'università ho fatto al massimo lettura da file. Non e' che mi potete consigliare uno schema logico da seguire.
Al max provare a svilupparlo insieme. Grazie

MaTz!
http://forum.ioprogrammo.it/thread.php?threadid=4913&boardid=20&styleid=1

Risponde johnkoenig
Ecco l'esempio...

```
#include "stdafx.h" // Per Visual C++
#include <stdio.h>
#include <io.h>
#include <time.h>
void main( void ) {
    struct _finddata_t c_file;
    long hFile;
/* Cerca il primo file nella directory corrente */
    if( (hFile = _findfirst( "*.*", &c_file )) == -1L)
       printf( "Non ci sono file nella
                              cartella corrente!\n" );
    else {
       printf( "Lista di tutti i file\n\n" );
       printf( "\nRDO HID SYS ARC DIR  FILE
                         DATE %25c SIZE\n", ' ' );
       printf( "--- --- --- --- ---  ----
                              ---- %25c ----\n", ' ' );
       printf( ( c_file.attrib & _A_RDONLY ) ?
                                   " Y  " : " N  " );
       printf( ( c_file.attrib & _A_HIDDEN ) ?
                                   " Y  " : " N  " );
       printf( ( c_file.attrib & _A_SYSTEM ) ?
                                   " Y  " : " N  " );
       printf( ( c_file.attrib & _A_ARCH )  ? "
                                    Y  " : " N  " );
       printf( ( c_file.attrib & _A_SUBDIR ) ? "
                                    Y  " : " N  " );
       printf( " %-12s   %.24s  %9ld\n",
       c_file.name, ctime( &( c_file.time_write
                                 ) ), c_file.size );
          /* Cerca tutti gli altri file */
          while( _findnext( hFile, &c_file )
                                          == 0 ) {
          printf( ( c_file.attrib & _A_RDONLY
                             ) ? " Y  " : " N  " );
          printf( ( c_file.attrib & _A_HIDDEN
                             ) ? " Y  " : " N  " );
          printf( ( c_file.attrib & _A_SYSTEM
                             ) ? " Y  " : " N  " );
          printf( ( c_file.attrib & _A_ARCH )
                               ? " Y  " : " N  " );
          printf( ( c_file.attrib & _A_SUBDIR
                             ) ? " Y  " : " N  " );
          printf( " %-12s %.24s  %9ld\n",
                             c_file.name, ctime( &(
              c_file.time_write ) ), c_file.size );
       }
       _findclose( hFile );
   }
}
```

Naturalmente, tutto dipende da dove inizi la ricerca (cartella di partenza).
Per il resto puoi studiarti un po' la struct *_finddata_t*

Riconosce le parole e le restituisce all'applicazione

Tu parli Java risponde

Impariamo a utilizzare Sphinx4, una tecnologia che ultimamente ha avuto una forte diffusione soprattutto per la sua capacità di riconoscere e analizzare la voce

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
J2SE

Software
J2SE SDK, Sphinx4

Impegno

Tempo di realizzazione

Per riconoscimento vocale si intende la possibilità da parte di un computer capire le parole pronunciate dall'utente. Grazie ad una grammatica che definisce le parole riconoscibili, un programma per il riconoscimento vocale analizza il flusso audio ricevuto e restituisce le parole che crede di aver capito. Questa è una tecnologia che ultimamente ha avuto una forte diffusione, soprattutto per quanto riguarda call center e servizi di informazioni automatizzati. Esistono chiaramente diverse implementazioni, anche hardware, per il riconoscimento vocale e noi andremo a vedere come sia semplice utilizzare un'implementazione software per Java, Sphinx4.

SPHINX4: UN RICONOSCITORE VOCALE IN JAVA

Sphinx4 è un riconoscitore vocale scritto totalmente in Java, basato su HMM (*Hidden Markov Models*), un modello statistico usato per il riconoscimento, e deriva dal predecessore Sphinx3. Nato da un progetto universitario, Sphinx4 si contraddistingue per essere il primo riconoscitore vocale OpenSource (cosa non da trascurare in questo campo dell'informatica). Si può definire un vero e proprio framework per lo sviluppo di applicazioni vocali, vista la grande quantità di API e tool disponibili. Sphinx4 permette il riconoscimento di due diverse modalità: live e batch. Live quando il programma analizza dei flussi audio generati in quell'istante, mentre la modalità batch permette l'analisi di un file wav contenente un flusso audio registrato precedentemente. Attraverso una configurazione che deve essere fornita a Sphinx4, quest'ultimo riesce a riconoscere determinate parole e a restituirle all'applicazione. Questo riconoscitore permette inoltre di avere un punteggio di riconoscimento della parola, ovvero una stima di sicurezza per la parola appena riconosciuta. Inoltre vengono date diverse possibilità per definire la grammatica, che definisce il linguaggio da riconoscere, come *JSGF (Java Speech Grammar Format),* formato di specifica delle grammatiche della SUN. Chiaramente l'approccio ad un tool con tantissime feature come Sphinx4 può risultare un po' ostico, soprattutto per quanto riguarda terminologie e aspetti noti soltanto a persone che hanno già avuto a che fare con riconoscitori vocali e tecnologie analoghe. Infatti all'interno di questo framework è presente un file di configurazione abbastanza complesso, che deve essere editato per decidere quali componenti si vogliono utilizzare e quali no, quali proprietà settare etc etc. Nonostante ciò noi vedremo insieme la realizzazione di un semplice programma Java che utilizza le potenzialità di Sphinx4 in maniera molto semplice, senza dover essere degli esperti del riconoscimento vocale.

CONFIGURAZIONE

Prima di poter utilizzare Sphinx4 è necessario capire la configurazione che necessita questo tool. Ogni programma che utilizza il riconoscimento vocale attraverso questo framework deve necessariamente avere un file di configurazione, un file dove sono presenti tutti i componenti che si devono utilizzare, dove vengono settate diverse proprietà importanti. Una grande limitazione è data dal fatto che non possiamo riconoscere direttamente parole italiane, visto che il dizionario di default è stato fatto per parole inglesi. La definizione di un nuovo dizionario non è nulla di trascendentale, ma sicuramente è poco pratica da spiegare in un articolo. Quindi ci adatteremo ad utilizzare un dizionario inglese presente nella distribuzione ufficiale di Sphinx4. Il dizionario da utilizzare viene richiamato da Sphinx4 nel file di configurazione nel seguente modo:

```
<component name="dictionary" type=
    "edu.cmu.sphinx.linguist.dictionary.FastDictionary">
<property name="dictionaryPath" value="resource:
 /edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_1
    6k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/
model/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz
                      _6800Hz/dict/cmudict.0.6d"/>
<property name="fillerPath"  value="resource:
 /edu.cmu.sphinx.model.acoustic.WSJ_8gau_13dCep_1
    6k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/
model/acoustic/WSJ_8gau_13dCep_16k_40mel_130Hz
                         _6800Hz/dict/fillerdict"/>
<property name="addSilEndingPronunciation"
                                    value="false"/>
<property name="allowMissingWords" value="false"/>
<property name="unitManager" value="unitManager"/>
</component>
```

Come potete vedere le dichiarazioni che vengono fatte sono molto intuitive. Viene definito il componente dizionario (*dictionary*) e vengono settate diverse proprietà del dizionario, prima fra tutte la locazione. Quindi se dovessimo cambiare dizionario (già all'interno di Sphinx4 ne sono presenti diversi per diversi scopi) dovremmo ritoccare questa definizione nel file di configurazione del nostro programma. Altra parte che non possiamo tralasciare riguarda la definizione della grammatica che verrà utilizzata nel nostro programma. Questa viene definita sempre come un componente e quindi inserita nella configurazione che dobbiamo fare

```
<component name="jsgfGrammar" type=
                    "edu.cmu.sphinx.jsapi.JSGFGrammar">
  <property name="dictionary" value="dictionary"/>
  <property name="grammarLocation" value="."/>
  <property name="grammarName" value="slideshow"/>
  <property name="logMath" value="logMath"/>
</component>
```

Il nome della grammatica, che definiamo di tipo JSGF (è possibile definire anche grammatiche in altri formati), è *"slideshow"* quindi nella directory del nostro programma oltre al file di configurazione dovremo avere il file *"slideshow.gram"*. Questi due settaggi sono obbligatori per sviluppare un'applicazione, visto che comunque dovremo definire una nostra grammatica e richiamarla in un nostro file di configurazione. Gli altri componenti che non vengono qui riportati richiedono un più attento studio di caratteristiche del riconoscimento, che in questo sede non sono di nostro interesse.

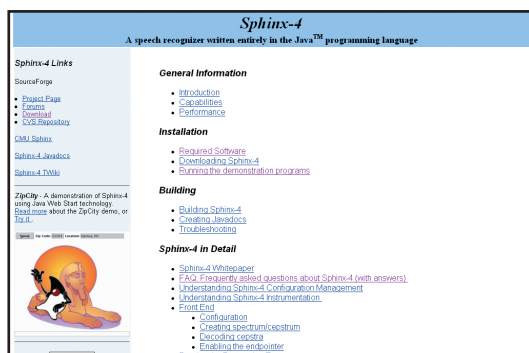



Fig. 1: **L'homepage di Sphinx4**

ESEMPIO DI APPLICAZIONE

Vediamo ora come poter sviluppare una nostra applicazione, abbastanza semplice, per usufruire di questo framework. Il programma che vogliamo sviluppare è un semplice SlideShow di immagini, che può essere azionato attraverso dei comandi vocali. Una cosa del genere potrebbe essere utile per le presentazioni o per le lezioni dove il relatore può tranquillamente essere lontano dal computer che proietta le immagini da presentare. Il funzionamento è abbastanza banale: dobbiamo permettere al nostro programma di riconoscere i comandi *"Next"* e *"Back"* per poter rispettivamente andare avanti e indietro nella nostra presentazione di immagini. Inoltre possiamo prevedere di far visualizzare ulteriori informazioni riguardanti la slide richiamando il comando *"About"*. Quindi prima di tutto dobbiamo definire una nostra grammatica, in base alla quale il nostro programma potrà capire quale comando gli è stato dato.

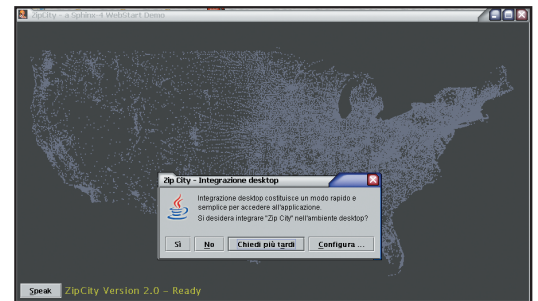



Fig. 2: **Un programma d'esempio con Java Web Start, disponibile sul sito ufficiale**

DEFINIZIONE DELLA GRAMMATICA

Per la definizione della grammatica utilizzeremo JSGF. Questo è uno standard della SUN che viene utilizzato per descrivere le grammatiche in programma di riconoscimento vocale. Praticamente all'interno di un semplice file vengono definiti dei comandi che poi verranno interpretati dal riconoscitore. Nel nostro caso dobbiamo definire pochi comandi, ecco quindi di seguito la grammatica che utilizzeremo.

```
#JSGF V1.0;
/**
 * slideshow.gram: Grammatica da utilizzare nel
                          programma SlideShow vocale */
grammar slideshow;
```

Il riconoscimento e la sintesi vocale sono dei campi di ricerca molto attivo in questi tempi. Sono molte le specifiche e i prodotti che sono presenti ora, molti anche i tool per sviluppatori.

http://www.research.att.com/projects/tts/demo.html

http://www.voiceobjects.com/eng/products/index.shtml

http://www.loquendocafe.com/

```
public <comando> = ( next | back | about );
public <uscita> = (exit | quit);
```

In questo modo abbiamo definito due diversi comandi. Il primo lo utilizzeremo per manovrare lo SlideShow, mentre il secondo per uscire dall'applicazione. Abbiamo definito i comandi in inglese perché, come detto precedentemente, per utilizzare delle parole in italiano dovremmo scrivere un dizionario italiano per Sphinx4, che va oltre le finalità di questo articolo. Ora che la definizione della semplice grammatica è stata fatta, bisogna implementate la parte riguardante il vero e proprio riconoscimento.

RICONOSCIMENTO DEI COMANDI

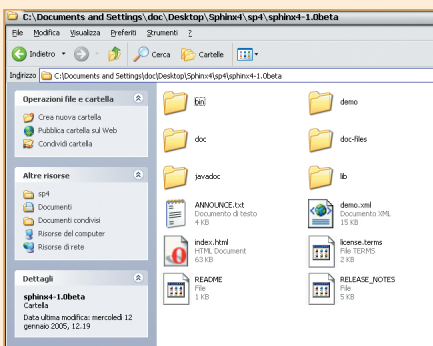
A questo punto dobbiamo utilizzare Sphinx4 per poter riconoscere il flusso audio esterno. Allora, prima di tutto, dobbiamo importare i package necessari per il nostro compito

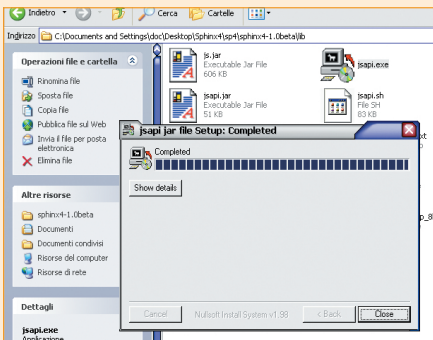
```
import edu.cmu.sphinx.frontend.util.Microphone;
import edu.cmu.sphinx.recognizer.Recognizer;
import edu.cmu.sphinx.result.Result;
import edu.cmu.sphinx.util.props.ConfigurationManager;
import edu.cmu.sphinx.util.props.PropertyException;
```

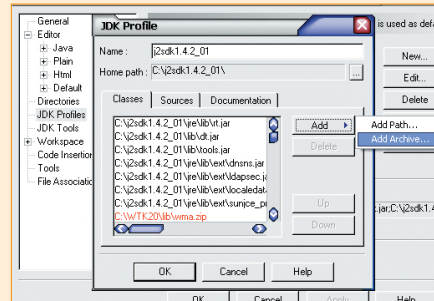
La prima cosa da fare ora è passare il file di configurazione a Sphinx4. Già abbiamo parlato di questo file di configurazione in XML, lungo e tedioso da compilare. Per evitare di perdersi in una lunga discussione immagineremo di avere a disposizione un nostro file di configurazione (in realtà lo trovate insieme al codice sul CD della rivista). Richiamiamo quindi il file di configurazione all'interno del nostro programma

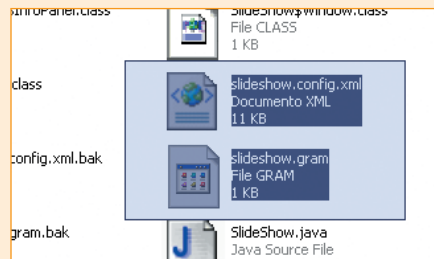
```
URL url = SlideShow.class.getResource(
```

SEI PASSI PER COMINCIARE

DOWNLOAD DI SPHINX4 E ESTRAZIONE FILE

1 Dopo aver scaricato la distribuzione di Sphinx4 dal sito http://cmusphinx.sourceforge.net/sphinx, scegliamo una cartella dove estrarre lo zip che crea una cartella con tutte le librerie, documentazione ed eseguibili organizzati.

SETUP DI JSAPI

2 All'interno della sottocartella lib troviamo l'eseguibile jsapi.exe (o jsapi.sh per Linux) che ci permette di installare JSAPI sul nostro sistema semplicemente avviandolo.

CLASSPATH

3 Per utilizzare Sphinx4 in un nostro programma dobbiamo avere inserito nel *classpath* i *jar* necessari. Questo possiamo farlo o definendolo a livello del nostro sistema, modificando la variabile *CLASSPATH* o definendolo all'interno del nostro ambiente di lavoro (ad esempio in *JCreator*). I *jar* da aggiungere sono presenti nella sottocartella *lib*.

SETUP DEL PROGETTO

4 Per utilizzare Sphinx4 in un nostro programma dobbiamo necessariamente definire un file XML di configurazione (trovate l'esempio incluso nel CD e nella distribuzione di Sphinx4) e un file per definire la grammatica delle parole da riconoscere.

INIZIALIZZAZIONE COMPONENTI

```
URL url = SlideShow.class.getResource(
                       "slideshow.config.xml");
ConfigurationManager cm =
                new ConfigurationManager(url);
Recognizer recognizer = (Recognizer)
                   cm.lookup("recognizer");
Microphone microphone = (Microphone)
                   cm.lookup("microphone");
recognizer.allocate();
```

5 Ora possiamo direttamente passare alla scrittura del codice, richiamando all'interno del nostro programma i componenti principali per il riconoscimento: Recognizer e Microphone.

EFFETTUARE IL RICONOSCIMENTO

```
if (microphone.startRecording())
{
  Result result = recognizer.recognize();
  if (result != null)
  {
     String resultText =
            result.getBestFinalResultNoFiller();
     System.out.println("Comando
           pronunciato: " + resultText + "\n");
  }
}
```

6 Concludiamo con il vero e proprio riconoscimento. In base alla grammatica da noi definita e al flusso audio che viene riconosciuto, la classe Result ci restituisce la stringa riconosciuta.

```
                                    "slideshow.config.xml");
ConfigurationManager cm = new ConfigurationManager(url);
```

Ora il Manager delle configurazioni di Sphinx4 conosce cosa vogliamo utilizzare. Infatti solo ora possiamo istanziare i diversi tool che dovremo utilizzare durante il nostro programma. In questo semplice caso dobbiamo avere a disposizione soltanto un oggetto Microphone, che rappresenterà appunto il microfono del nostro computer, e un oggetto *Recognizer,* che riconoscerà il flusso audio in base anche alla grammatica che abbiamo definito precedentemente. Per istanziare questi due oggetti dobbiamo appunto passare attraverso il ConfigurationManager come illustrato qui di seguito

```
Recognizer recognizer = (Recognizer)
                              cm.lookup("recognizer");
Microphone microphone = (Microphone)
                              cm.lookup("microphone");
```

Istanziato il riconoscitore e il microfono dobbiamo procedere nel seguente modo: prima dobbiamo essere certi che il microfono stia registrando, poi dobbiamo dire al riconoscitore di processare il flusso audio. In questo modo lasciamo in attesa il riconoscitore fino a quando non trova una parola conosciuta, in tal caso effettuiamo l'operazione richiesta

```
recognizer.allocate();
if (microphone.startRecording()) {
   System.out.println("Microfono in registrazione.");
   while (true) {
      Result result = recognizer.recognize();
      if (result != null) {
         //Adesso controlliamo il risultato ottenuto dal
         //riconoscitore attraverso la classe Result e ci viene
         //restituito il risultato che ha la maggiore affidabilità
         //in base alla grammatica che abbiamo definito
         String resultText =
                   result.getBestFinalResultNoFiller();
         System.out.println("Comando pronunciato: " +
                                    resultText + "\n");
         //In questo punto dovremo effettuare uno switch
         //sui vari comandi presenti nel nostro programma
         //e in base al comando riconosciuto avviare una
         //determinata funzione
      } else {
         System.out.println("Comando non pronunciato");
} } } }
else {
   System.out.println("Impossibile avviare la
                                    registrazione.");
   recognizer.deallocate();
   System.exit(1); }
```

Attraverso queste semplici righe di codice abbiamo quindi il controllo di cosa viene detto dall'utente e quindi rimane soltanto da implementare il vero e proprio programma di SlideShow.

GUI

Per quanto riguarda le parte grafica della nostra applicazione utilizzeremo un Layout grafico che fa al caso nostro: *CardLayout.* Questo layout, che fa parte di *AWT (Abstract Widget Toolkit),* permette di inserire un certo numero di pannelli che vengono richiamati e mostrati a schermo in una certa sequenza. Praticamente quando noi salviamo dei pannelli in questo layout diamo anche una stringa che identifica questo pannello. Successivamente quando vorremo visualizzare un determinato pannello *CardLayout* si comporterà come una tabella di *Hash:* passando semplicemente la stringa che identifica il pannello lo potremo richiamare e visualizzarlo a schermo.

Fig. 3: Lo schema dell'applicazione

Quindi nel programma che stiamo costruendo utilizzeremo questo componente, decidendo (dinamicamente o staticamente) quali sono i file da dover visualizzare ed inserendoli in questo layout. Poi successivamente dovremo scrivere la parte di controllo del nostro programma che, in base alle parole riconosciute da Sphinx4, manovrerà adeguatamente i vari pannelli. Allora prima di tutto definiamo come saranno strutturati i vari pannelli che faremo visualizzare al nostro programma. Dobbiamo suddividerli in pannelli contenenti una slide e pannelli contenenti approfondimenti (quindi testuali). Nel pannello contenente una slide dovremo semplicemente stampare a schermo l'immagine che viene passata nel costruttore

```
class ImmPanel extends JPanel {
   Image image;
   public ImmPanel(Image image) {
      this.image = image;
      this.setBorder(BorderFactory.createTitledBorder(
         BorderFactory.createEtchedBorder( Color.black,
          Color.blue ),"SlideShow by IoProgrammo") );}
   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawImage(image,0,0,this.getWidth(),
                              this.getHeight(),this);} }
```

Per quanto riguarda il pannello di approfondimento richiameremo dinamicamente dei file testuali e mostreremo un'informazione sulla diapositiva tramite *JOptionPane,* ovvero visualizzando un semplice

L'IBM è una società molto attiva nel campo del vocale. Ultimamente ha dichiarato che rilascerà il suo riconoscitore vocale come progetto opensource. Sul sito di AlphaWorks sono disponibili molti tool per lo sviluppo di applicazioni vocali.
http://www.alphaworks.ibm.com/tech/voiceportal

http://www.alphaworks.ibm.com/tech/embeddedvoicetk

NOTA

JSGF *(Java Speech Grammar Format)* è un formato che permette di definire la grammatica da utilizzare in applicazioni vocali. È stato definito all'interno di JSAPI (Java Speech API), specifica della SUN per lo sviluppo di applicazioni vocali.
http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/
http://java.sun.com/products/java-media/speech/

PopUp. Quindi la classe che implementeremo servirà per caricare il file delle informazioni aggiuntive della diapositive e successivamente per richiamarle all'interno del programma

```
class Info{
  String file,linea;
  Vector info=new Vector();
  public InfoPanel(String file) {
     this.file=file;
     try {
        BufferedReader br=new BufferedReader(new
          InputStreamReader(new FileInputStream(new
                                          File(file))));
        while ((linea=br.readLine())!=null)
        info.add(linea);}
     catch (Exception e) {
        info.add("Nessuna informazione trovata");}}
  public String getAbout(int i) {
     return (String)info.elementAt(i-1);}}
```

Ora che abbiamo completato la definizione dei vari pannelli da vedere dobbiamo pensare alla parte di programma che farà da ponte tra il comando che viene riconosciuto e quello che dobbiamo visualizzare.

CONTROLLER

Rimane soltanto da inserire il motore alla nostra applicazione. Per fare ciò sono sufficienti poche righe di codice. Abbiamo già il risultato del riconosci mento vocale, ottenuto tramite la classe *Result* di Sphinx4. Questa classe ci restituisce, analizzando il flusso audio, la parola che viene pronunciata. A questo punto, avendo definito poche e semplici comandi dovremo soltanto implementare uno *switch* e di volta in volta eseguire l'operazione appropriata. Ecco quindi lo *switch* da inserire nel nostro programma

Fig. 5: ***Il risultato che otteniamo.***

```
if (resultText.equals("next")) {
  cardManager.next(ImPanel);
  visualizzanda++;}
else if (resultText.equals("back")) {
  cardManager.previous(ImPanel);
  visualizzanda--; }
else if (resultText.equals("about")) {
  String news=info.getAbout(visualizzanda);
  JOptionPane.showMessageDialog(sl,news); }
else if (resultText.equals("exit"))
System.exit(0);
else if (resultText.equals("quit"))
System.exit(0);
```

Così, tutta l'applicazione è ai comandi della nostra voce, facendo andare avanti e indietro le slide, mostrando info aggiuntive e uscendo dal programma.

TEST DELL'APPLICAZIONE

Per avviare la nostra applicazione dobbiamo essere sicuri di avere bene configurato il microfono sul nostro computer, altrimenti il nostro programma non potrà riconoscere niente. Una volta configurato dobbiamo includere nel nostro classpath le librerie di Sphinx4, che troviamo nella cartella lib della distribuzione scaricabile dal sito di Sphinx4 *(http://cmusphinx.sourceforge.net/sphinx4/)*. All'interno della cartella *lib* troviamo anche l'eseguibile per installare JSAPI sul nostro sistema, jsapi.exe per Windows, jsapi.sh per Linux. Una cosa molto importante per l'esecuzione corretta del programma è il passaggio del parametro *"-mx312m"* alla Virtual Machine di Java. Con questo parametro riserviamo abbastanza memoria al nostro programma. Infatti facendo partire lo stesso programma senza questo parametro andremo sicuramente ad incappare in un errore di memoria. Nella directory del nostro programma dovremo inoltre inserire i file della presentazione. Per questo programma sono stati definiti staticamente nel codice, ma possiamo anche immaginare di far partire il programma passando come parametro una directory e ricavare dinamicamente i file che servono (immagini e testo). Una volta avviata dobbiamo soltanto godere di questo gioiellino tecnologico, andando avanti e indietro tra le slide.

CONCLUSIONI

Le applicazioni che possiamo sviluppare grazie al riconoscimento vocale sono tantissime. Ad esempio, pur non avendo alcun tool relativo al suo interno, Sphinx4 potrebbe essere la base di partenza per scrivere un programma per il riconoscimento vocale delle persone, attraverso lo spettro della voce. Inoltre se pensiamo ad un nostro programma che utilizza Sphinx4 insieme ad un sintetizzatore vocale, come ad esempio FreeTTS, vediamo un'applicazione che capisce cosa viene detto e magari risponde. Il punto dolente è la definizione di un dizionario personalizzato, per il quale vi rimando al sito di Sphinx4 dove c'è un'esauriente documentazione al riguardo. Per il resto buon divertimento con questo utile (e soprattutto opensource) framework.

Federico Paparoni

Oltre a Sphinx4, esistono altre implementazioni di riconoscitori vocali. Ecco un paio di link di altri famosi prodotti da poter utilizzare
http://www.cloudgarden.com/
http://www.lhs.com/
http://www.microsoft.com/speech/

Applicazioni multilingua con PHP

In che lingua te lo dico?

Viviamo nell'era globale. Le nostre applicazioni soprattutto se orientate al Web devono risultare portabili non solo da un sistema all'altro ma anche da una lingua all'altra, scopriamo come fare!

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Principi di PHP

Software
PHP 5

Impegno

Tempo di realizzazione

Inutile che io dica a voi, esperti navigatori di Internet, che viviamo nell'era del villaggio globale. Ormai lo sapete benissimo, non conta se sviluppate per l'Italia, per l'America e perché no per la Cina! Conta invece quanto la vostra applicazione può essere facilmente utilizzata in ciascuno di questi paesi senza che voi dobbiate riprogrammarla per renderla fruibile in ciascuna delle lingue parlate in questi paesi. Detto in parole povere, se volete avere una minima possibilità di diffondere le vostre applicazioni al di fuori del mercato locale, dovrete progettarle in modo che siano localizzabili in qualsiasi lingua. Detto questo, e senza dilungarci troppo, nei grandi vantaggi economici/commerciali di cui può godere un'applicazione pronta per essere facilmente esportata, è il caso di capire come questo prodigio possa avvenire.

CONOSCETE IL CINESE?

Ho il dubbio che molti programmatori non conoscano il tedesco, allo stesso modo ho il dubbio che una buona parte non conosca neanche l'Inglese, volendo poi esaminare qualcuna delle altre possibilità sconfineremmo nella certezza, qualcuno conosce il Cinese? Detto questo è importante stabilire che non dobbiate essere voi a tradurre le vostre applicazioni, ma che piuttosto debba essere reso disponibile a chiunque una sorta di Kit che, in modo semplice e senza per questo dover conoscere una riga di programmazione, metta in grado di tradurre un'applicazione senza doverla riprogrammare/ricompilare o in qualche modo mettere mano al codice.
Riepilogando, la nostra applicazione multilingua dovrà essere composta in tre strati

Applicazione Base -> interprete -> vocabolario

molto semplicemente l'interprete sarà una mini-applicazione che prenderà le stringhe che caratterizzano la versione base, le confronterà con un vocabolario contenente stringhe in lingua straniera e produrrà un output nella lingua opportuna.

Applicazione Base contiene la stringa: "Questo articolo è fantastico"
Vocabolario Italiano contiene: "Questo articolo è fantastico"->"This article is beatiful"

L'interprete controlla se nel vocabolario inglese c'è una stringa opportuna rispetto a quella base e se c'è la utilizza per fornire all'utente la versione nella lingua desiderata. Il meccanismo sembra molto semplice ed ha alcuni interessanti vantaggi. Infatti posto che venga diffuso un vocabolario delle stringhe base, chiunque ne fosse capace potrebbe produrre le corrispondenti stringhe in una lingua straniera. Basterebbe poi settare una qualche variabile di confi-

DA DOVE INIZIARE

PHP deve essere compilato con il supporto a GetText. Se installate PHP da una distribuzione linux, potete semplicemente scaricare il modulo corretto da uno dei repository della distribuzione. Viceversa in ambiente windows potete semplicemente decommentare il commento che riguarda gettext all'interno del PHP.ini. E' importante dotarsi di tutto il software necessario per creare dei vocabolari. Al solito in ambiente linux tutto é presente in modo abbastanwa autonomo, semplicemente installando il pacchetto che contiene gettext nella vostra distribuzione. In ambiente Microsoft potete scaricare e installare tutto da *http://gettext.sourceforge.net*

gurazione nel programma, per mostrare l'interfaccia del software nella lingua desiderata. Va da se che a questo punto sorge qualche piccolo problema. Il primo problema riguarda la scrittura dell'interprete. Il secondo problema riguarda il formato del vocabolario. Chiaramente l'interprete e il vocabolario sono due strumenti molto legati fra loro. L'interprete deve conoscere infatti il formato in cui è strutturato il vocabolario per poterlo utilizzare. A questo punto si possono seguire due strade:

1. Ideare un proprio formato di vocabolario personalizzato e poi programmare un apposito interprete
2. Utilizzare un qualche formato specifico possibilmente standard e sfruttare un interprete adeguato già esistente per questo formato.

poiché siamo amanti degli standard noi seguiremo questa seconda strada. Prima di parlare di come sarà fatto il nostro vocabolario, è però corretto definire alcune proprietà che vorremmo che avesse. In particolare il nostro vocabolario deve essere poter utilizzato da chiunque conosca una lingua straniera diversa da quella in cui abbiamo programmato l'applicazione, dunque deve essere semplice! Non necessariamente sarà usato da un programmatore, piuttosto sarà usato da un traduttore, perciò dovremo fornirgli una comoda applicazione facile facile da usare, senza troppi fronzoli. In secondo luogo il nostro vocabolario dovrà essere portabile. Dovrà essere utilizzato sia con Windows che con Linux. Infine vorremmo che il nostro vocabolario non sia legato alla scelta di un linguaggio di programmazione. In realtà semplicemente cambiando l'interprete ma lasciando invariato il formato del vocabolario, esso potrebbe essere utilizzato con applicazioni scritte in PHP ma anche in Delphi, o in Visual Basic, C++ e quant'altro. Per tutti questi motivi, la nostra scelta ricade su un prodotto OpenSource chiamato GetText.

SIETE PRONTI PER GETTEXT?

GetText è un progetto elaborato da GNU, perciò decisamente OpenSource. Tutta la documentazione sul progetto è reperibile all'indirizzo *http://www.gnu.org/software/gettext/*. Quasi tutto il software OpenSource multilingua in circolazione sfrutta *GetText*. Il vocabolario utilizzato da applicazioni progettate secondo *GetText* è un normale file di testo! Il che lo rende estremamente portabile e utilizzabile da chiunque. È anche vero che nel tempo sono state sviluppate applicazioni molto semplici per gestire in modo visuale un vocabolario *GetText*, come ad esempio *PoEdit – http://www.poedit.com* – per cui è facilmente intuibile che un eventuale traduttore può utilizzare queste applicazioni per generare o modificare un vocabolario *GetText*. Infine *GetText* è utilizzabile sia sotto Linux che sotto Windows, e con tutti o quasi tutti i linguaggi attualmente disponibili. In questo articolo utilizzeremo PHP in ambiente Linux, e di volta in volta spiegheremo anche come e dove reperire i tool per compiere gli stessi passi in ambiente Windows.

IL PRIMO PROGRAMMA PHP

Al solito il miglior modo per capire qualcosa di quello che fin qui abbiamo accennato in maniera teorica è dare uno sguardo ad un esempio:

```
$language="it_IT";
putenv("LANG=$language");
setlocale(LC_ALL,$language);
$domain='messages';
bindtextdomain($domain,"/home/jaco/languages");
textdomain($domain);
$stringa=gettext("Hello World");
echo $stringa;
```

La prima funzione interessante di questo esempio è la funzione *"setlocale()"*. Con questa funzione si può indicare a PHP che alcune particolari stringhe devono essere formattate secondo una lingua piuttosto che un'altra. Considerate ad esempio il classico:

```
echo strftime("%A %e %B %Y", mktime(
                                0, 0, 0, 03 27, 2005));
```

restituisce: *Sunday 27 March 2005*

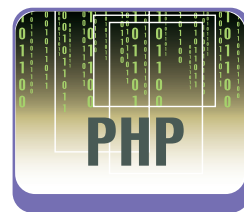
invece la forma:

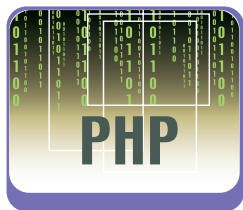
```
$language="it_IT";
setlocale(LC_ALL,$language);
echo strftime("%A %e %B %Y", mktime(
                                0, 0, 0, 03, 27, 2005));
```

restituisce: *domenica 27 marzo 2005*

Cioè con le stringhe correttamente formattate in italiano. A questo punto seguono alcune righe decisamente più interessanti

```
$domain='messages';
bindtextdomain($domain,"/home/jaco/languages");
textdomain($domain);
$stringa=gettext("Hello World");
```

con la seconda e terza riga informiamo PHP che i nostri vocabolari saranno contenuti in una gerarchia al di sotto del percorso */home/jaco/languages*,

IL SOFTWARE POEDIT

Manipolare i vocabolari é abbastanza semplice. D'altra parte come abbiamo già visto si tratta di semplici file di testo, tuttavia se volete dotare i vostri traduttori di un'applicazione che li astragga completamente dal formato utilizzato nei vocabolari, potete allora usare poedit, un software specifico con il formato utilizzato da gettext e reperibile all'indirizzo *http://sourceforge.net/projects/poedit/*

inoltre i file che contengono i vocaboli sarnno chiamati *messages.po*. Infine con la funzione *gettext* si avvia il processo di traduzione. *Gettext* controlla se nel vocabolario *it_IT* definito in *setlocale* c'è una stringa corrispondente a *Hello World*, se c'è riempie il contenuto della variabile *$stringa* con la traduzione adeguata. Il meccanismo non è complesso. Si tratta veramente di poche istruzioni. Perciò il nostro problema adesso è creare il vocabolario.

CREARE IL VOCABOLARIO

Il punto di partenza è sempre il file index.php esattamente come l'abbiamo creato in precedenza, e cioè:

```
<?
$language="it_IT";
setlocale(LC_ALL,$language);
echo strftime("%A %e %B %Y", mktime(
                                0, 0, 0, 03, 27, 2005));
$domain='messages';
bindtextdomain($domain,"/home/jaco/languages");
textdomain($domain);
$stringa=gettext("Hello World");
echo $stringa;
?>
```

a questo punto usiamo il comando *xgettext* come segue:

```
xgettext -n index.php
```

se siete in ambiente Linux dovreste avere già tutto installato nel momento in cui avete installato il pacchetto contenente *gettext*, viceversa se siete in ambiente windows, trovate tutto quello che vi occorre alla url *http://gettext.sourceforge.net*. Il risultato dell'esecuzione di questo comando è un file *message .po*. Un normale file di testo che contiene le seguenti definizioni:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT
                                        HOLDER
# This file is distributed under the same license as the
                              PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-02-28 11:01+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
#: index.php:8
msgid "Hello World"
msgstr ""
```

Le ultime due righe sono esplicative. Forniamo una traduzione per l'italiano modificando il file come segue:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT
                                        HOLDER
# This file is distributed under the same license as the
                              PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-02-28 11:01+0100\n"
"PO-Revision-Date: 2005-02-28 11:13+0100\n"
"Last-Translator: jaco <jaco@jaco.it>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
#: index.php:8
msgid "Hello World"
msgstr "Ciao Mondo"
```

Ed anche in questo caso le ultime due righe sono significative. Ci mancano ancora due passi per finalizzare il tutto. Il primo passo è compilare il file *messages.po*. Niente di più facile, il comando è:

```
msgfmt messages.po
```

questo comando da luogo a un file messages.mo che altro non è che la versione compilata di *messages.po*. L'ultimo passo è creare una struttura su file system in grado di ospitare i file fin qui creati.
Ricordiamo che abbiamo usato:

```
bindtextdomain($domain,"/home/jaco/languages");
```

quindi va da se che questa struttura che il nostro punto di partenza sarà */home/jaco/languages*.
Il resto della struttura va creato secondo la regola generale *<DOMAIN>/<LANG_CODE>/LC_MESSAGES/*.mo* dove *<DOMAIN>* è il percorso che abbiamo dichiarato richiamando la funzione *bindtextdomain*. Per cui la struttura delle directory che ci consente di usufruire delle funzioni offerte da *gettext* è

APPROFONDIMENTI

GETTEXT E IL SOFTWARE OPEN SOURCE

GetText é il metodo usato dalla maggior parte dei software OpenSource oggi disponibili. La maggior parte delle applicazioni Open Source sono scritte utilizzando GetText come sistema per la gestione delle lingue. Data la grande diffusione dell'OpenSource si tratta di un indicatore piuttosto valido dell'utilità di questyo sistema.

la seguente:

```
/home/jaco/languages/it_IT/LC_MESSAGES/messages.mo
```

dove *messages.mo* è il file che abbiamo compilato in precedenza. Provate a puntare il browser adesso su *http://localhost/index.php* – adeguando ovviamente al percorso utilizzato sul vostro webserver – e otterrete:

domenica 27 marzo 2005
Ciao Mondo

tuttavia è sufficiente modificare *index.php* come segue

```
<?
$language="en";
setlocale(LC_ALL,$language);
echo strftime("%A %e %B %Y", mktime(
                              0, 0, 0, 03, 27, 2005))."<br>";
$domain='messages';
bindtextdomain($domain,"/home/jaco/languages");
textdomain($domain);
$stringa=gettext("Hello World");
echo $stringa;
?>
```

per ottenere di nuovo

Sunday 27 March 2005
Hello World

COMPLICHIAMOCI UN PO LA VITA

Creiamo un secondo file paperino.php composto come segue:

```
<?
$language="it_IT";
setlocale(LC_ALL,$language);
$domain='messages';
bindtextdomain($domain,"/home/jaco/languages");
textdomain($domain);
$stringa=gettext("ioProgrammo is the best magazine
                                        of the world");
echo $stringa;
?>
```

Copiamo il messages.po che abbiamo ottenuto in precedenza in un file *old.po* ed eseguiamo *xgettext -n paperino.php*, otteniamo un file *messages.po* composto come segue:

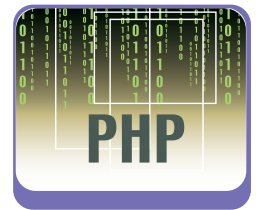
```
# SOME DESCRIPTIVE TITLE.
[...]
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-02-28 12:01+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
#: paperino.php:7
msgid "ioProgrammo the best magazine of the world"
msgstr """
```

possiamo motificarlo aggiungendo la traduzione *"ioProgrammo la migliore rivista del mondo"* nel campo *msgstr* ma a questo punto ci troviamo di fronte ad un enigma. Abbiamo cioè due file, old.po e messages.po ognuno ha dentro delle stringhe che possono servirci, ma quale dei due usare? Dobbiamo trovare un modo di unificare i due file senza perdere le modifiche fin qui ottenute. Niente di più facile.
Il comando da usare è :

```
msgmerge messages.po old.po –output-file=new.po
```

questo comando genera un file che miscela i due ottenuti in precedenza, e il cui contenuto è il seguente:

```
[...]
# This file is distributed under the same license as the
                                          PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-02-28 11:01+0100\n"
"PO-Revision-Date: 2005-02-28 12:03+0100\n"
"Last-Translator: jaco <jaco@jaco.it>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
#: index.php:8
msgid "Hello World"
msgstr "CIao Mondo"
msgid "ioProgrammo the best magazine of the world"
msgstr "ioProgrammo la migliore rivista del mondo"
```

non ci resta che compilare il file appena ottenuto con *msgfmt new.po* e copiare il file messages.mo così ottenuto nella directory corrispondene ed il gioco è fatto.
Otterremo un unico vocabolario che serve due file *.php*

LE FONTI UFFICIALI

GetText é un sistema piuttosto complesso che tiene conto di un gran quantitativo di variabili, che vanno dalla gestione dell'output delle date a quello dei caratteri speciali. Non era ovviamente possibile trattare tutti questi aspetti all'interno di questo articolo, che tuttavia rimane sufficiente per iniziare a sviluppare le vostre applicazioni multilingua. Il sito ufficiale dove reperire informazioni specifiche in relazione a qualche particolare problema é: http://www.gnu.org/software/gettext

Come costruire applicazioni web utilizzando STRUTS

Java e Struts i belli del Web

Vi spieghiamo come creare applicazioni Web conformi al modello MVC che ci consente di separare la logica dell'applicazione dalla sua grafica di presentazione

Utilizza questo spazio per le tue annotazioni

Struts è un nome di battesimo decisamente originale per un framework; è impossibile quindi non chiedersi: "come si pronuncia?", ma soprattutto: "cosa significa? ". Alla prima domanda non vi è risposta certa. Nel mio ufficio l'ho sentito pronunciare nei modi più originali: *"struz, straz"*, anche se la fonetica esatta pare sia *straz*. La seconda domanda invece, ha una risposta decisamente meno "randomica", Babylon in merito al suo significato suggerisce: "si pavoneggia, andature impettite". A quanto pare, abbiamo a che fare con un framework impronunciabile ma decisamente superbo! Per esperienza acquisita nell'utilizzo, vi assicuro che la sua superbia è del tutto fondata; utilizzare Struts presenta moltissimi vantaggi e pochissimi svantaggi, tutto questo a costo zero. Bene, dopo questa breve parentesi introduttiva, quasi pubblicitaria oserei dire, proseguiamo la nostra lettura alla scoperta di Struts, e soprattutto prepariamoci a costruire la nostra prima applicazione web basata su di esso.

COME INIZIARE

Prima di tutto c'è bisogno di un JDK recente. Poi ovviamente c'è bisogno del *framework struts*. Infine c'è bisogno dell'*Application server Tomcat*. Un componente opzionale è l'ambiente di sviluppo eclipse. Trovate tutto nel CD allegato alla rivista, o ovviamente sui rispettivi siti d'origine. Diamo per scontato che abbiate installato sia il JDK che Tomcat. L'installazione di struts è relativamente semplice, è sufficiente scompattare il file contenente il pacchetto e copiare tutti i file *.jar* nella directory *WEB-INF/*, i file .tld andranno invece nella directory *WEB-INF/lib*. Sarà necessario poi configurare il file *web.xml* e il file *struts-config.xml* secondo una certa logica che tratteremo in modo approfondito all'interno dell'articolo. Molto brevemente il file *web.xml* conterrà le informazioni relative a chi dovrà servire le richieste dell'applicazione, mentre il file *struts-config.xml* conterrà informazioni relative a come la richiesta debba essere gestita.

COS'È STRUTS E SOPRATTUTTO A COSA SERVE

Struts è un framework gratuito sviluppato dalla Apache Jakarta Project Group, che consente di utilizzare al meglio, all'interno di una applicazione web (basata su J2EE), il pattern *MVC (model, view, controller)*.

COS'È E A COSA SERVE IL PATTERN MVC

Il pattern MVC viene utilizzato per separare, all'interno di una generica applicazione, la parte di *Business Logic (Model)*, la parte di *Interfaccia Utente (View)* e la parte di *Program Progression (Controller)*. Esso è composto da tre componenti principali:

- **Model:** il modello, si occupa della definizione delle informazioni, quindi rappresenta i dati nel sistema.
- **View:** La vista, rappresenta l'interfaccia utente, come i dati sono rappresentati all'utente.
- **Controller:** Il "meccanismo di controllo", si occupa di verificare il flusso e lo stato dei dati inseriti dall'utente. Favorisce lo scambio di dati tra *Model* e *View*.

Ad oggi, l'utilizzo del pattern MVC è molto diffuso ed

i vantaggi che presenta sono effettivamente numerosi:

- Consente di separare la parte di business rules dalla presentation, favorendo così la modularità dei componenti.
- Chi sviluppa il front-end *(view)*, non ha bisogno di sapere come queste pagine verranno popolate (mediante il *model*).
- È possibile, mediante il *Controller*, gestire la visualizzazione dei dati in maniera dinamica. *View* diverse per utenti diversi, più *View* per lo stesso utente, ecc..
- È possibile centralizzare la sicurezza dell'applicazione all'interno del *Controller*. Con il pattern MVC, l'interfaccia verso l'utente viene fatta passare solo attraverso l'oggetto *controller*, questo rappresenta quindi l'unico punto d'accesso e come tale l'unico punto in cui vanno effettuati tutti i controlli di sicurezza.

STRUTS E IL CONTROLLER

Il *controller*, come già abbiamo accennato, all'interno di un'applicazione MVC svolge diverse funzioni, tra queste ha l'onere di ricevere input dal client, invocare una o più operazioni appartenenti alla logica dell'applicazione e gestire la risposta da restituire al client. In *Struts*, gli oneri nel controller, sono gestiti tramite vari componenti, uno di questi è rappresentato da un'istanza della classe *org.apache.struts.action.ActionServlet*. La suddetta classe, estende la classe *javax.servlet.http.HttpServlet* e si occupa di gestire, a fronte di una richiesta HTTP, il giusto reindirizzamento all'interno del framework.
Detto in soldoni riceve una richiesta da parte di un client e stabilisce chi la deve gestire a seconda della tipologia di richiesta.
È importante sapere che va configurata all'interno del file: *web.xml* (descrittore di deploy di una generica applicazione web). La sua configurazione verrà descritta in seguito.
Riassumendo, la classe *ActionServlet*, "decide" (a fronte di una richiesta HTTP), quale classe dovrà soddisfare la richiesta effettuata dal client. La classe "prescelta", viene definita helper; questa sa esattamente quale operazione deve svolgere per soddisfare la richiesta del client. Nel framework *Struts*, questa classe deriva dalla classe *org.apache.struts.action .Action*.

IL FILE WEB.XML

Sebbene questo file sia generico per la configurazione di tutte le applicazioni web, ci sono delle opzioni di configurazione specifiche per *Struts*. Un esempio di file *web.xml* potrebbe essere il seguente:

```
<web-app id="StrutsWebApp">
<servlet>
    <servlet-name>action</servlet-name>
<servle-class>
    org.apache.struts.action.ActionServlet
</servlet-class>
</servlet>
</web-app>
```

Con queste righe di configurazione abbiamo detto che tutte le richieste HTTP fatte all'applicazionedevono essere gestite dalla classe *org.apache.struts.action.ActionServlet*. Questa classe come già detto assolverà al ruolo di controller.

```
<web-app id="StrutsWebApp">
<servlet>
</servlet>
<servlet-mapping>
<servlet-name>action</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
</web-app>
```

Queste righe indicano che solo le richieste effettuate a pagine con estensione *.do devono essere gestite dalla classe *org.apache.struts.action.ActionServlet*. Utilizzando questo approccio si possono eventualmente fare convivere in una stessa web application pagine gestite con *struts* e pagine gestite in modo tradizionale.

LA CLASSE ACTIONSERVLET

Rappresenta un'estensione del componente *Controller*. Il suo compito è quello di "mettere in moto" la logica di Business dell'applicazione a fronte di una richiesta client. La classe *Action* può svolgere diverse funzioni e contenere diversi metodi, ma il quello più importante in assoluto, è: *execute()*. Questo viene richiamato dal controller a fronte di una request proveniente dal client. È importante sapere che questo metodo restituisce un oggetto di tipo *org.apache.struts.action.ActionForward*.
Esso determina una destinazione verso la quale il controller può inviare il controllo al termine della *Action*. I forward delle azioni, come le *Action* da richiamare, vengono definiti nel file di configurazione messo a disposizione da struts: *struts-config.xml*.
Un esempio di file config.xml potrebbe essere il seguente:

```
<action path="/showListaProdotti" type=
```

COS'È UNA WEB APPLICATION

Una Applicazione web basata su J2EE è un insieme di servlet, jsp, pagine html, classi ed altre risorse che collegate tra di loro, creano una applicazione web completa Questa può essere installata ed eseguita (in maniera concorrente) da web container multipli. Per motivi "contestuali" la definizione di applicazione web è poco esaustiva, per chi avesse ancora dei dubbi si consiglia uno studio un po' più approfondito mediante altre fonti.
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

```
        "web.ShowListaProdottiAction" scope="session">
...........................
</action>
```

Abbiamo "mappato" l'azione */showListaProdotti* con la Action *web.ShowListaProdottiAction.* Questo significa che ogni volta che il controller riceverà una request che nell'URI contiene la stringa *"/showListaProdotti"* verrà richiamato il metodo *execute()* della classe *web.ShowListaProdottiAction.*

```
<action path="/showListaProdotti" type=
        "web.ShowListaProdottiAction" scope="session">
  <forward name="error" path="/Home.jsp"></forward>
  <forward name="ok" path=
                        "/showListaProdotti.jsp"></forward>
</action>
```

L'azione *"/showListaProdotti"* definisce due forward: *<forward name="error">* e forward: *<forward name="ok">*, questo significa che: se il metodo *execute()*, restituisce un *ActionForward = "error"* l'applicazione visualizzerà la pagina: *"Home.jsp"*, se invece restituisce un *ActionForward = "ok"* L'applicazione visualizzerà la pagina *showListaProdotti.jsp.*
Riepilogando, nel file *web.xml* viene specificato che tutte le richieste di una pagina *.do* devono essere gestite dalla classe *struts-config.xml* e richiama il metodo *execute()* della classe preposta a servire la richiesta. Il metodo *execute()* restituisce un forward che può essere utilizzato per passare il controllo a una pagina terza, nel nostro caso è stato utilizzato per produrre un output di errore o esito positivo.
Tutta questa procedura riassume il concetto di *"Controller".* Le azioni eseguite dalle classi che gestiscono le richieste esauriscono invece il concetto di *"Model".*

IL MODELLO

La collocazione di questo componete in Struts (ma in generale un tutte le applicazioni web), non è così banale. In generale, il confine che separa gli oggetti della logica di Business da quelli della presentazione è sempre molto sottile e varia a seconda delle esigenze di ogni singola applicazione. In ogni caso, la "regola" della buona programmazione, suggerisce di mantenere gli oggetti *Business* della logica dell'applicazione, separati dalla presentazione. Il motivo è pressoché ovvio, in questo modo, se si decide di cambiare la grafica dell'applicazione ci risparmiamo di cambiare anche gli oggetti di *Business!* In Struts il componente *model,* si può identificare con un'istanza della classe: *org.apache.struts.action.ActionForm.* Gli oggetti *ActionForm,* hanno la stessa funzione che può avere un comune oggetto *formBean,* e cioè, si occupano di passare i dati dallo strato *client,* allo strato *business.* Come avviene questo passaggio di dati? Il framework preleva automaticamente i dati dalla *request,* li passa alla *Action* utilizzando un oggetti di tipo *ActionForm.*
Il mapping tra *Action* e *ActionForm* viene sempre definito nel file di configurazione: *struts-config.xml.* In pratica data una form HTML dobbiamo capire come il passaggio dei dati viene effettuato verso la action che esaudirà la richiesta. Per farlo abbiamo bisogno di creare una classe che estenda la *ActionForm,* recuperi i dati e una volta validati li spedisca alla action che deve gestirli. Il tutto viene fatto ancora una volta nel file *struts-config.xml,* ad esempio:

```
<action path="/showListaProdotti" name="
                  ShowListaProdottiForm" scope="session">
</action>

<form-beans>
<form-bean name="ShowListaProdottiForm"
                  type="web.ShowListaProdottiAction">
  </form-bean>
</form-beans >
```

LA VISTA

Per questo ultimo componente, in realtà, non c'è molto da dire, come in una qualsiasi applicazione web basata su J2EE, anche in Struts si identifica mediante: pagine HTLM, Servlet, JSP, tag Custom ecc..
In **Figura 1** è mostrato dove si colloca Struts in una tipica applicazione web *Tree tiers.*

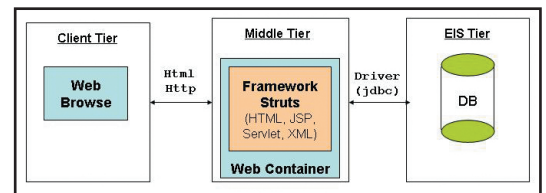



***Fig. 1:* Dove si colloca Stuts all'interno di una applicazione Tree tiers**

IL CARRELLO DELLA SPESA

Arrivati a questo punto, i veri programmatori, saranno già stanchi di tutte queste chiacchiere e saranno sicuramente ansiosi di cominciare a sviluppare. Bene, come non accontentarli?
Partiamo quindi con lo sviluppo della nostra prima applicazione web basata su Struts: *"Carrello della spesa"* Se vi piace il suo nome siete già a buon punto!!
Cosa ci consente di fare questa applicazione?
Beh, dal suo nome si evince che ci consente di comprare qualcosa. Sarà presente una prima schermata iniziale dove è possibile scegliere i prodotti d'acqui-

stare. Mediante dei pulsanti di navigazione è possibile compiere un'intera procedura d'acquisto:

- **1.** - Scegli i prodotti e inseriscili in un carrello virtuale.
- **2.** - Visualizza il carrello e conferma l'acquisto.
- **3.** - Inserisci i dati per l'Acquisto.
- **4.** - ACQUISTA!!

IL WORK FLOW DELL'APPLICAZIONE

La nostra applicazione: *"Carrello della spesa"* è composta da diversi "componenti" e la sua struttura può essere riassunta in: **Figura 2**.

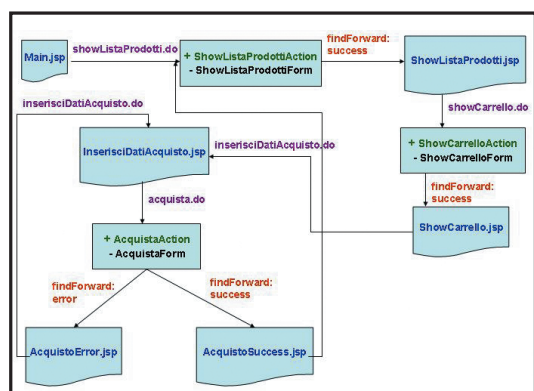



***Fig. 2:* Struttura dell'applicazione web**

Tenendo ben presenta questo diagramma, cominciamo ad analizzare il work flow di Struts in relazioni ai suoi componenti.
La prima Action che viene richiamata all'avvio dell'applicazione, è la *initial.do*, questa a sua volta richiama la *index.do* che effettua un forward diretta alla *Main.jsp*.
Quindi Ricapitolando:

```
startApplication -> initial.do -> index.do  -> Main.jsp
```

Siamo così arrivati alla prima jsp *(Main.jsp)* presente nel nostro diagramma di riferimento.
Questa pagina presenta una sintassi molto semplice, il suo unico compito è quello di creare il frameset che conterrà la nostra applicazione, e di richiamare (all'interno di un frame) la *showListaProdotti.do.*
Di seguito riportiamo un "pezzo" significativo della pagina in questione:

```
<frameset>
    <frame name="header" src="showListaProdotti.do">
</frameset>
```

Facile no?? Rispecchia perfettamente quanto detto sopra!! Ma proseguiamo la nostra analisi, scoprendo quale cataclisma si verifica nel modo web quando si richiama la: *showListaProdotti.do.*

```
<action path="/showListaProdotti" type=
                        "dae.ShowListaProdottiAction" name=
                                "showListaProdottiForm" scope=
                                    "session" validate="false">
<forward name="success" path=
                                    "/ShowListaProdotti.jsp">
</forward>
</action>
```

Il frammento di *struts-config.xml* relativo alla action *showListaProdotti,* ci dice che:

La *showListaProdotti.do,* crea una istanza (se già non esiste) della *showListaProdottiForm,successivamente* richiama la *ShowListaProdottiAction* che con un forward *name="success"* richiama la *ShowListaProdotti.jsp.* Anche questa operazione descritta in tre righe appare decisamente semplice, ma addentriamoci nei componenti Struts in questione, e vediamo cosa accada al loro interno.

Componente model

Il modello *ShowListaProdottiForm* rappresenta uno dei model dell'applicazione, al sui interno viene semplicemente istanziato un ArrayList che conterrà tutti i prodotti disponibili per l'acquisto.

Componente controller

All'interno del controller *ShowListaProdottiAction,* precisamente nel metodo *execute(),* viene:

- Recuperata un'istanza dell'*ActionForm.*
- Creata, mediante la classe *CarrelloDB(),* la lista dei prodotti disponibili per l'acquisto.
- Popolato l'ArrayList dello *ShowListaProdottiForm,* con i prodotti disponibili per l'acquisto.
- Eseguito il findForward alla pagina: *ShowListaProdotti.jsp*

Ecco l'implementazione del metodo:

```
public ActionForward execute(ActionMapping mapping,
            ActionForm form, HttpServletRequest request,
                            HttpServletResponse response)
{ ShowListaProdottiForm prodottiForm =
                                (ShowListaProdottiForm)form;
  CarrelloDB model = CarrelloDB.get();
  ArrayList prodotti = model.creaProdotti();
  prodottiForm.setListaProdotti(prodotti);
  return mapping.findForward("success");
}
```

È importante notare che:

- Non si crea una nuova istanza dello *ShowListaProdottiForm,* ma viene preso e "castato" quello

passato al metodo *execute()*. Il "tipo specifico" di form e quello definito nello *struts-config.xml.*

- La classe *CarrelloDB()*, è la classe dove avvengono tutte le operazioni di creazione e salvataggio dei dati. Per questa ultima funzionalità può vagamente rappresentare una simulazione di un DB.

LA VISTA

Abbiamo gia detto che se il metodo *execute* va a buon fine, viene richiamato con un forward il file *ShowListaProdotti.jsp.* Si tratta della pagina dove compare la lista dei prodotti disponibili per l'acquisto e dove l'utente può scegliere i prodotti da aggiungere nel suo carrello. La sua implementazione riporta alcuni tag (facenti parte della grande famiglia Struts HTML tag) interessanti, me vediamoli da vicino:

```
<html:form action="acquista" method="post">
```

html:form, è il degno sostituto del HTML *<form>*. Tra i sui diversi attributi, uno dei più importanti è sicuramente: *action.* Questo consente di definire quale action richiamare nel metodo *submit()* del form.

```
String acquistaProdottoVal = "value(acquistaProdotto-"
                                  + codiceProdotto + ")";
<html:checkbox  property=
                         "<%=acquistaProdottoVal%>" />
```

html:checkbox, anche lui sostituisce HTML *<input> checkbox*, l'attributo property rappresenta il nome dell'attributo passato al momento della *request.*
In questo caso, è definito come la concatenazione di più stringhe questo perché Struts consente di definire nel *FormAction* una generica *HashMap* contenente diversi valori contrassegnati da un key da noi definita, nel nostro caso la key è rappresentata dal *codiceProdotto.* I valori memorizzati nelle *HashMap*, posso essere prelevati, mediante questa key.
L'*HashMap* è molto utile quando si devono passare liste di valori ma naturalmente, l'attributo *property* può anche essere semplicemente una Stringa, un int ecc..

Lista Prodotti

Codice Prodotto	Descrizione Prodotto	Prezzo Prodotto	Prodotto Disponibile	Acquista Prodotto
001	Fotocamera	450		
002	Lettore DVD	500		
003	Televisore	1000		
004	PC	450		
005	Web-Cam	100		
006	Scanner	100		
007	Tastiera PC	50		
009	Tostapane	35		
0010	SONY PS2	199		

Aggiungi al Carrello

Fig. 3: Pagina jsp che visualizza i prodotti

```
<html:submit styleClass="invia" property="Submit">
                                   Acquista</html:submit>
```

html:submit, sostituisce HTML *<input> submit* e rispetto ad esso, non presenta particolari differenze. Il suo look and feel della pagina *ShowListaProdotti .jsp* visibile in: **Figura 3**.
Scelti i prodotti d'acquistare, l'utente prosegue la sua navigazione fino ad arrivare all'operazione PAGA. La pagina *ShowCarrello.jsp* mostra il contenuto del carrello, visibile in **Figura 4**.

Dati Carrello

Codice Carrello: 1
Data Creazione: 12/02/2004

Dettaglio prodotti aggiunti nel Carrello

Codice prodotto	Descrizione prodotto	Prezzo Prodotto
001	Fotocamera	450
003	Televisore	1000
009	Tostapane	35

Totale Ordine: 1485

Torna alla Lista | Inserisci dati per l'Acquisto

Fig. 4: Pagina jsp che mostra il contenuto del carrello

Le istruzioni:

```
<bean:write name="showCarrelloForm" property=
                                       "idCarrello"/>
<logic:iterate name="showCarrelloForm" property=
                  "listaProdotti"   id="singoloProdotto">
```

Appartengono entrambi alla famiglia *Struts Logic Tags.* In particolare, *bean:write*, consente di scrivete all'interno di una pagina *jsp,* il valore dell'attributo specificato nel *property,* prelevandolo dal FormAction specificato nel name. Il Tag logic:iterate, consente invece di iterare un' attributo di tipo collection. Ad esempio, combinato al tag *bean:write*, consente di stampare in una pagina *jsp*, tutti i valori di una lista. (Questa operazione, è visibile tra i sorgenti della pagina in questione).
La pagina *InserisciDatiAcquisto.jsp* è usata per l'inserimento dei dati d'acquisto, visibile in **Figura 5**.

```
<html:html><html:html>
<html:text property="nome"></html:text>
<html:select property="tipoCarta">
```

***Fig. 5:* Pagina jsp per l'inserimento dei dati d'acquisto**

```
<html:option value="visa">Visa</html:option>
<html:select>
<html:button styleClass="invia" property="Submit"
                    onclick="showListaProdotti();">
                  Torna alla Lista</html:button>
```

Anche questi tags, fanno parte dei tag HTML Struts, e non fanno altro che ridefinire i loro omonimi in HTML.

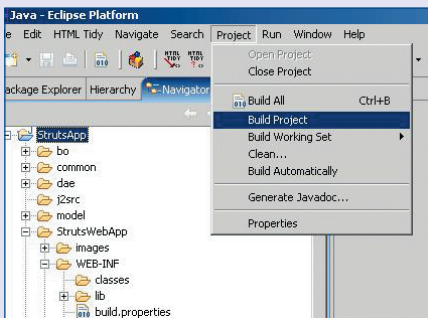
CONCLUSIONI

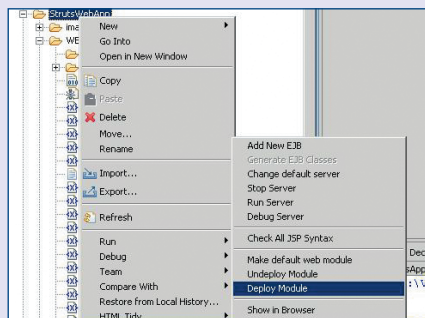
Dagli esempi riportati nell'articolo, ma soprattutto tra i sorgenti dell'applicazione, avete scoperto, ma soprattutto scoprirete (almeno spero), tanti aspetti interessanti del framework Struts. Le sue potenzialità sono decisamente notevoli, degne di un nome così superbo!! Il mio carrello l'ho riempito… vi invito a riempire il vostro. Buona Spesa

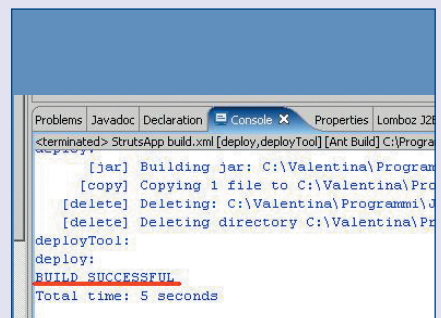
Valentina Muraglia

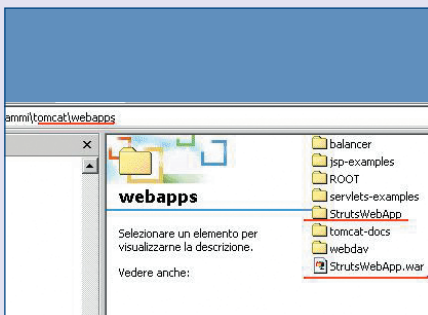
http://jakarta.apache.org/struts/userGuide

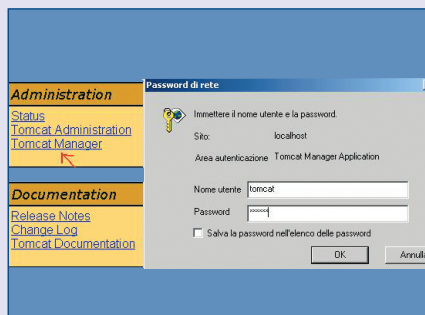
AUTOMATIZZARE LO SVILUPPO CON ECLIPSE E LOMBOZ

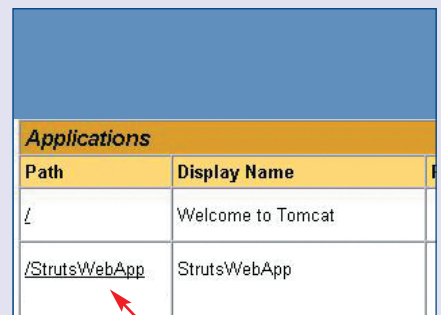
COMPILIAMO

1 **Apriamo eclipse e posizioniamoci sotto la root del progetto *StrutsApp*. Da questa posizione apriamo il menu a tendina *Project* e lanciamo il comando *Build Project, per iniziare il nostro lavoro.***

DEPLOY

2 **Posizioniamoci sotto la dir *StrutsWebApp,* col il tasto destro del mouse selezioniamo la voce di menu *Lomboz* J2EE, si aprirà un nuovo menù, da questo selezioniamo il comando *Deploy Module*.**

ESITO DEPLOY

3 **Dalla finestra *Console* di eclipse verifichiamo l'esito del deploy. Accertiamoci che alla fine compaia: *BUILD SUCCESSFUL!!* Questo ci garantisce che l'applicazione è stata correttamente inviata a Tomcat.**

VERIFICHIAMO IL DEPLOY SU TOMCAT

4 **Posizioniamoci sotto la directory: … *tomcat\webapps* e verifichiamo che ci siano sia la dir: *StrutsWebApp* che il file *StrutsWebApp.war*. Se ci sono… il deploy è terminato!**

LANCIAMO TOMCAT

5 **Lanciamo Tomcat con il comando *"localhost:8080"*, per default il localhost gira sulla porta 8080, selezioniamo il comando *Tomcat Manager* con password = *tomcat.***

LANCIAMO L'APPLICAZIONE

6 **A questo punto, comparirà la lista di tutte le applicazioni deployate su tomcat da qui selezioniamo *StrutsWebApp* e il gioco è fatto!**

Programmazione parallela con Java

Thread e blocchi critici

Per scrivere programmi "seri" è necessario conoscere un po' di programmazione parallela. La breve serie che inizia questo mese spiega come si usano i thread nei programmi Java

Utilizza questo spazio per le tue annotazioni

È capitato a tutti di usare programmi che non sono bravi a fare più cose contemporaneamente. Schiacci un tasto che lancia una lunga operazione... e tutto si ferma. Magari vuoi interrompere l'operazione, ma non si può più schiacciare niente finché non è finita da sola. Alcuni programmi non aggiornano nemmeno lo schermo, quindi basta spostare la finestra per ottenere un inquietante macchione bianco.

E che dire dei sistemi client-server? Quasi sempre, un server deve servire più client contemporaneamente.

Se apro un'applicazione multiutente, non posso certo aspettare che tutti gli utenti connessi prima di me siano usciti dal sistema. Anche in questo caso, il programma non può fare le cose "in sequenza". Per fortuna i computer ci permettono di ottenere l'illusione del parallelismo. Nello stesso programma, più sequenze di istruzioni possono girare "in parallelo", come se avessero ciascuna una CPU dedicata. Queste sequenze di istruzioni si chiamano thread. In realtà questo parallelismo è un'illusione: la maggior parte dei computer hanno una sola CPU.

Quest'unica CPU esegue una sola sequenza di istruzioni in ogni dato istante, ma è molto brava ad alternare velocemente le sequenze. Dando a ciascun thread un pezzo del suo tempo (time sharing), il computer fa in modo che nessun thread si senta abbandonato, e che tutti i thread procedano apparentemente di pari passo.

I thread sono delle bestiole capricciose, ma non se ne può fare a meno: in molte situazioni, un programma "single thread" non è accettabile.

Per fortuna Java supporta direttamente i thread, senza bisogno di librerie esterne. Quindi fare programmazione parallela in Java è più facile che nella gran parte degli altri linguaggi.

LA CORSA DELLE LUMACHE

Per introdurre i thread scriveremo un breve simulatore di corsa di lumache:

```
public class Lumaca {
    private final String _nome;

    public Lumaca(String nome) {
        _nome = nome;
    }
    public void corriFinoAlTraguardo() {
        System.out.println(toString() + ": 1 metro");
        System.out.println(toString() + ": 2 metri");
        System.out.println(toString() + ": arrivata!");
    }
    public String toString() {
        return _nome;
    }
}
```

Il metodo *corriFinoAlTraguardo()* stampa il progresso della lumaca, che su un computer moderno è molto più veloce di una lumaca reale: tre metri di distanza in meno di un millisecondo.

```
public class CorsaDiLumache {
    public static void main(String[] args) {
        Lumaca[] lumache = creaLumache();
        gareggia(lumache);
    }
```

```
    private static void gareggia(Lumaca[] lumache) {
        for (int i = 0; i < lumache.length; i++)
            lumache[i].corriFinoAlTraguardo();
    }
    private static Lumaca[] creaLumache() {
        return new Lumaca[] {
                new Lumaca("Speedy"),
                new Lumaca("Fulmine"),
                new Lumaca("Furia")
        };
    }
}
```

CorsaDiLumache crea tre lumache con nomi appropriati e le fa gareggiare una dopo l'altra. Il risultato non è particolarmente interessante:

Speedy: 1 metro
Speedy: 2 metri
Speedy: arrivata!
Fulmine: 1 metro
Fulmine: 2 metri
Fulmine: arrivata!
Furia: 1 metro
Furia: 2 metri
Furia: arrivata!

Per rendere le cose più divertenti, le lumache dovrebbero correre insieme, "in parallelo". Per farlo, ci servono i thread.
Se vogliamo far girare un pezzo di programma in parallelo con altri pezzi di programma, dobbiamo implementare l'interfaccia *java.lang.Runnable*. Questa interfaccia definisce il metodo *run()*, che deve contenere il codice parallelo. Aggiorniamo la nostra Lumaca per farla diventare un Runnable:

```
public class Lumaca implements Runnable...
    public void run() {
        corriFinoAlTraguardo();
    }
```

Ora dobbiamo lanciare questo Runnable in parallelo con il resto del codice. Per farlo dobbiamo creare un *java.lang.Thread*. Uno dei costruttori di Thread prende appunto un *Runnable*. Poi possiamo lanciare l'esecuzione parallela con il metodo *Thread .start()*:

```
public class CorsaDiLumache...
    private static void gareggia(Lumaca[] lumache) {
        for (int i = 0; i < lumache.length; i++) {
            Thread t = new Thread(lumache[i]);
            t.start();
        }
    }
```

Thread.start() fa una serie di operazioni di basso livello che creano un nuovo thread nell'ambito dello stesso processo. A un certo punto, *start()* chiamerà il metodo *run()* del Runnable con cui è stato inizializzato il thread, e la Lumaca farà la sua breve corsa. A differenza di un normale metodo, Thread *.start()* non aspetta che il metodo *Runnable .run()* sia terminato, ma ritorna subito al programma principale. Quindi, mentre la Lumaca parte, *CorsaDiLumache* crea altre due lumache e le fa partire nello stesso modo.
Ecco il risultato:

Fulmine: 1 metro
Speedy: 1 metro
Furia: 1 metro
Fulmine: 2 metri
Speedy: 2 metri
Furia: 2 metri
Fulmine: arrivata!
Furia: arrivata!
Speedy: arrivata!

In generale, ogni volta che il programma gira, il risultato cambia. Il computer e la Java Virtual Machine garantiscono che i thread contenenti le lumache possono girare in parallelo, ma non fanno promesse per quanto riguarda l'ordine e la frequenza con cui la CPU è allocata ai vari thread. Nel momento in cui usiamo i thread in un programma, il programma smette di essere deterministico e acquista una componente di casualità. La cosa è accentuata dal fatto che diverse Virtual Machine usano algoritmi diversi per gestire i thread, quindi un comportamento "comune"

TROPPA PRUDENZA

Ricorda che synchronized fa l'esatto contrario di quello che un thread vorrebbe fare: dove il codice vorrebbe essere parallelo, synchronized lo rende nuovamente seriale. Questo significa che se sincronizzi troppo poco, ti esponi a bug molto pericolosi - ma se sincronizzi troppo, rischi che il tuo codice non sia affatto parallelo! Nelle prossime puntate di questo corso cercheremo una giusta via di mezzo per risolvere questo problema.

THREAD E PROCESSI

Quando si parla di parallelismo, spesso si fa confusione tra thread e processi. I thread e i processi hanno parecchi punti in comune, ma sono cose molto diverse. Chiunque possegga un sistema operativo moderno usa regolarmente il parallelismo. Basta lanciare più programmi, o lanciare più istanze dello stesso programma. Questi flussi di esecuzione si chiamano task o processi, e sono gestiti dal Sistema Operativo. Sono quelli che appaiono nel Task Manager di Windows. I processi vivono come isole in aree di memoria diverse e controllate dal sistema, quindi un processo non può creare facilmente problemi in un altro processo. Per lo stesso motivo, però, è difficile comunicare tra processi. Di solito si usano meccanismi "lenti" come la rete o i file. I thread sono flussi di esecuzione nell'ambito dello stesso processo (nel caso di Java, la stessa Virtual Machine), e condividono lo stesso spazio di memoria.
Sono molto meno sicuri dei processi, perché due thread possono facilmente pestarsi i piedi a vicenda.
In cambio, è molto facile condividere le informazioni tra thread: in un programma Java, i thread possono semplicemente condividere gli stessi oggetti

su una JVM può diventare "raro" sull'altro. Una cosa interessante da notare è che il *main()* termina presumibilmente prima della fine del programma. Dopo aver creato e lanciato i vari thread, *CorsaDiLumache.main()* ritorna al chiamante (la JVM). Ma il programma non termina finché non sono terminati i metodi *run()* di tutte e tre le Lumache. In generale e con alcune eccezioni, un programma non termina fin quando non è terminato l'ultimo thread. Non possiamo dare quasi niente per scontato quando lavoriamo con i thread. Non è nemmeno detto che la Lumaca che parte per prima sia anche la prima che stampa qualcosa sullo schermo. Nell'esempio reale trascritto qui sopra, la prima lumaca che arriva al metro è Fulmine, anche se il thread di Speedy è stato lanciato per primo. Il programma ha fatto in tempo a creare e lanciare Fulmine mentre il thread di Speedy non era ancora arrivato alla prima stampa. Ma accade più spesso che sia Speedy ad arrivare per prima, quindi questa privilegiata mette a buon frutto i suoi decimi di millisecondo di vantaggio. Le operazioni di stampa sembrano alternarsi tra le tre lumache, ma anche questa è una questione di statistica. Mi è capitato ad esempio che una lumaca tirasse una volata fantastica:

Fulmine: 1 metro
Fulmine: 2 metri
Fulmine: arrivata!

BLOCCARE I THREAD

Con la parola chiave synchronized possiamo dichiarare che un metodo è "atomico". Nessun thread può chiamare questo metodo se un altro thread l'ha chiamato e ne deve ancora uscire.

```
class LaMiaClasse {
    public synchronized
        void faiQualcosa() {
        ... }
}
```

Se creo un'istanza di LaMiaClasse e la condivido tra più thread, ho comunque la garanzia che solo un thread alla volta potrà invocare il metodo *faiQualcosa()* su quell'oggetto. Un metodo synchronized non è più "parallelo", ma "seriale". Se un thread chiama *faiQualcosa()* mentre un altro thread lo esegue, il secondo arrivato resterà bloccato in attesa fino a quando la risorsa non sarà libera. Quando la risorsa si libera, il thread viene sbloccato e può entrare nel metodo. Se più thread sono in attesa, la JVM assegna la risorsa a uno dei thread (non non necessariamente quello che aspetta da più tempo) in base al suo algoritmo preferito. La "*sincronia*" delle risorse si basa sul concetto di *lock*. Un metodo (o un altro blocco di codice) ha un "lucchetto" associato. Un thread chiude il lucchetto quando entra nel metodo, e lo apre quando ne esce. Se il lucchetto è chiuso, chi arriva aspetta fuori. La chiusura del lucchetto, l'esecuzione del codice e l'apertura del lucchetto sono un'unica operazione indivisibile. In Java tutti gli oggetti hanno un lock. Se chiamo un metodo synchronized su un oggetto (che è la mia risorsa condivisa), divento il temporaneo padrone del lock di quell'oggetto. Quindi altri thread possono tranquillamente chiamare lo stesso metodo su un altro oggetto della stessa classe, che ha un altro lock, ma non sullo stesso oggetto. Un'eccezione sono i metodi synchronized che sono anche static. In questo caso il lock non è quello dell'oggetto su cui viene chiamato il metodo, ma quello della sua classe (per essere precisi, quello dell'istanza di Class che la rappresenta). Quindi qualsiasi chiamata allo stesso metodo su qualsiasi oggetto della stessa classe è serializzata.

CODICE PARALLELO (IN CINQUE SEMPLICI PASSI)

```
public void X implements Runnable {
    ...
}
```

1 **Definisci una classe che implementa l'interfaccia Runnable**

```
public void X implements Runnable {
  public void run() {
    Utility.metodoStatico();
  }
}
```

2 **Nel metodo run() scrivi il codice che deve essere eseguito in parallelo. Puoi pensare alla classe come ad un programma, dove il metodo run() sostituisce il main().**

```
Thread t = new Thread(new X());
```

3 **Istanzia un oggetto della classe, e usalo per istanziare un oggetto di classe Thread.**

```
t.start();
```

4 **Lancia il thread chiamando il metodo start().**

```
public class Utility {
  public synchronized void metodoStatico() {
    ...
  }
}
```

5 **Controlla che il codice parallelo non utilizzi risorse che possono essere condivise da altri thread, come metodi statici o reference ad oggetti che arrivano "dall'esterno". Se questo avviene, usa synchronized per proteggere le risorse condivise**

Speedy: 1 metro
Speedy: 2 metri
Furia: 1 metro
Furia: 2 metri
Speedy: arrivata!
Furia: arrivata!

Anche i "sorpassi" nel bel mezzo della corsa sono abbastanza rari – ma ogni tanto se ne vedono, come in questo caso:

Speedy: 1 metro
Furia: 1 metro
Fulmine: 1 metro
Speedy: 2 metri
Fulmine: 2 metri
Furia: 2 metri
Speedy: arrivata!
Fulmine: arrivata!
Furia: arrivata!

Vai, Fulmine!

Leggere sullo schermo l'andamento della corsa è divertente, ma la cosa che ci interessa di più è il nome del vincitore. Con il tipo di output che abbiamo, non è molto comodo capire chi abbia vinto.
Sarebbe meglio annunciare esplicitamente un vincitore ufficiale. Per farlo ho introdotto una classe Arbitro:

```
public class Arbitro {
  private boolean _garaChiusa = false;
  public void segnalaArrivo(Lumaca l) {
    if(_garaChiusa)
    return;
    System.out.println("**** " + l + " VINCE!!! ****");
    _garaChiusa = true;
  }
}
```

Ora tutte le Lumache possono chiamare l'Arbitro quando arrivano al traguardo. La prima che lo fa viene annunciata come vincitrice. Le altre vengono ignorate, perché la gara è già chiusa. Dobbiamo modificare le Lumache in modo che conoscano e usino un Arbitro:

```
public class Lumaca implements Runnable...
  private final Arbitro _arbitro;
  public Lumaca(String nome, Arbitro a) {
    _nome = nome;
    _arbitro = a;
  }
  public void corriFinoAlTraguardo() {
    System.out.println(toString() + ": 1 metro");
    System.out.println(toString() + ": 2 metri");
    System.out.println(toString() + ": arrivata!");
    _arbitro.segnalaArrivo(this);
  }
```

Se non vogliamo proteste e ricorsi, è importante che le Lumache condividano tutte lo stesso arbitro:

```
public class CorsaDiLumache...
  public static void main(String[] args) {
    Arbitro a = new Arbitro();
    Lumaca[] lumache = creaLumache(a);
    gareggia(lumache);
  }
  private static Lumaca[] creaLumache(Arbitro a) {
    return new Lumaca[] {
        new Lumaca("Speedy", a),
        new Lumaca("Fulmine", a),
        new Lumaca("Furia", a)
    };
  }
```

La prima volta che ho fatto girare questo programma i risultati sono stati incoraggianti:

Fulmine: 1 metro
Speedy: 1 metro
Furia: 1 metro
Fulmine: 2 metri
Speedy: 2 metri
Fulmine: arrivata!
Furia: 2 metri
Speedy: arrivata!
***** Fulmine VINCE!!! *****
Furia: arrivata!

Furia e Speedy hanno fatto in tempo a fare un po' di strada dopo che Fulmine aveva già tagliato il traguardo, ma prima che l'Arbitro la annunciasse come vincitrice ufficiale.
Cose come queste sono normali nella programmazione parallela, quindi non me ne preoccupo. Quello che invece mi preoccupa sono i risultati come questo:

Speedy: 1 metro
Furia: 1 metro
Fulmine: 1 metro
Speedy: 2 metri
Fulmine: 2 metri
Furia: 2 metri
Speedy: arrivata!
Fulmine: arrivata!
****** Speedy VINCE!!! ******
****** Fulmine VINCE!!! ******
Furia: arrivata!

Proprio così, abbiamo due vincitrici! E in mol-

MAGIA NERA

Inutile farsi illusioni: la programmazione parallela è difficile. Gli errori sono spesso subdoli, a volte perfidi. Capita che un bug si manifesti casualmente solo una volta ogni tanto (in questo caso si dice che il bug non è riproducibile), o che si manifesti mentre il programma gira ma non durante il debugging. A volte, le sequenze di avvenimenti che portano ad un errore sono talmente complicate da essere del tutto imprevedibili. Visto che non si può fare a meno dei thread, l'unica opzione che rimane è quella di imparare bene ad usarli, e di essere sempre molto prudenti quando si condividono risorse nel codice parallelo.

ti esperimenti ne vengono fuori addirittura tre. Con tutta la simpatia per la salomonica decisione arbitrale, questo non è ciò che ci aspettavamo. Evidentemente il programma non funziona. Perché?
Il colpevole è *Arbitro.segnalaArrivo()*. Tutte e tre le Lumache chiamano questo metodo sullo stesso Arbitro. Ma ciascuna Lumaca vive in un thread separato, quindi capita che il metodo venga chiamato contemporaneamente da più thread. Si dice che l'Arbitro è una risorsa condivisa tra i thread. Le risorse condivise sono indispensabili se vogliamo che i thread interagiscano. Ma sono anche un problema, perché nel momento in cui più thread accedono alla stessa risorsa, possono accadere cose bizzarre.
Quello che avviene in questo caso è che una Lumaca chiama per prima il metodo. Il metodo controlla il valore di *_garaChiusa*, scopre che è *false* e prosegue con l'annuncio della vincitrice. In questo frangente, una seconda Lumaca chiama lo stesso metodo sullo stesso oggetto. Il flag *_garaChiusa* non è ancora diventato true, quindi anche questa seconda Lumaca viene dichiarata vincitrice. Poi entrambe le esecuzioni del metodo settano *_garaChiusa* a *true* – due volte, ma ormai troppo tardi. Le due (o tre) esecuzioni del metodo hanno già annunciato ciascuna una vincitrice diversa. Che fare?
Per evitare questo problema dobbiamo fare in modo che *segnalaArrivo()* diventi un'operazione atomica. Questo significa che vogliamo che solo un thread alla volta possa eseguire il metodo *segnalaArrivo()* sullo stesso Arbitro. Per fare questo possiamo usare la parola chiave *synchronized* (vedi box).

```
public class Arbitro...
    public synchronized void segnalaArrivo(Lumaca l)...
```

A pensarci bene, questa non è la prima volta che le Lumache condividono una risorsa sincronizzata: era già successo nella versione precedente del programma, durante la stampa sullo schermo. Evidentemente il metodo *System.out.println()* è sincronizzato. In caso contrario le Lumache potrebbero stampare contemporaneamente sullo schermo, creando il disastro che si può immaginare. Se ciascun thread può terminare serenamente di stampare una frase e andare a capo prima che un altro thread cominci a fare lo stesso, lo dobbiamo a qualche tipo di sincronizzazione.
A questo punto potremmo pensare di aver sistemato tutto. Ma come spesso accade quando si usano i thread, il nostro programma ha ancora un subdolo bug.
Ecco cosa è saltato fuori durante uno dei miei esperimenti:

Fulmine: 1 metro
Speedy: 1 metro
Furia: 1 metro
Fulmine: 2 metri
Speedy: 2 metri
Furia: 2 metri
Fulmine: arrivata!
Speedy: arrivata!
****** Speedy VINCE!!! ******
Furia: arrivata!

Potete immaginare la costernazione di Fulmine, che si è vista soffiare ingiustamente la vittoria sotto il naso. Cosa diavolo è successo, questa volta?
Il problema è che ciascuna Lumaca annuncia il proprio arrivo in due punti: prima lo stampa sullo schermo, poi lo segnala all'Arbitro. Ma tra le due operazioni può avvenire che un'altra Lumaca, in un altro thread, porti a termine entrambi gli annunci. Quindi la povera vincitrice, che ha già annunciato per prima sullo schermo di essere arrivata al traguardo, si vede superare dalla concorrente che arriva prima al cospetto dell'Arbitro – che dà sempre retta alla prima Lumaca che vede. Questo dovrebbe dare un idea di quanto la programmazione parallela possa diventare subdola.
Il mese venturo risolveremo anche questo problema.

Paolo Perrotta

SENZA RUNNABLE

Negli esempi abbiamo definito una classe che implementa *Runnable*, ne abbiamo creato un'istanza e l'abbiamo usata per costruire un Thread.
Esiste un modo più veloce per definire un thread: ereditare direttamente dalla classe *Thread*:

```
public class Lumaca extends Thread...
```

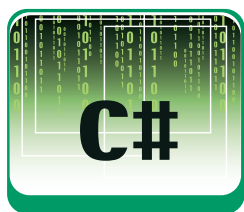
Il resto della Lumaca è identico alla versione già esistente. Anche in questo caso si devono mettere le operazioni parallele in un metodo *run()* (nota che la classe *Thread* implementa a sua volta *Runnable*). Per lanciare il thread si deve chiamare *start()*, non *run()*. Quest'ultimo è un normale metodo, e non lancia alcun thread separato. Sarà *start()* a fare il lavoro sporco e a chiamare a sua volta *run()*:

```
public class CorsaDiLumache...
    private static void gareggia(
                        Lumaca[] lumache)
{
    for (int i = 0; i < lumache.length; i++)
    lumache[i].start();
}
```

In questo modo è un po' più semplice lanciare i thread. Ciò detto, di solito è meglio implementare *Runnable* che ereditare da Thread.
Java non ha ereditarietà multipla, quindi una classe che eredita da Thread non può ereditare da nessun altro.

Aggiorniamo il sistema operativo e gli applicativi via codice

Sistema sempre aggiornato

Grazie a Windows Update Agent è davvero semplice usufruire dei servizi di aggiornamento offerti da Microsoft. In questo articolo parleremo delle tecniche per rilevare, scaricare ed installare patch

Studiando le novità introdotte dal Platform SDK per Windows XP SP2 si scopre un'interessante tecnologia: *Windows Update Agent*. Mediante le interfacce COM che costituiscono *Windows Update Agent* (WUA, d'ora in avanti) è possibile passare in rassegna tutte le patch presenti nel sistema ed eventualmente installare quelle mancanti.

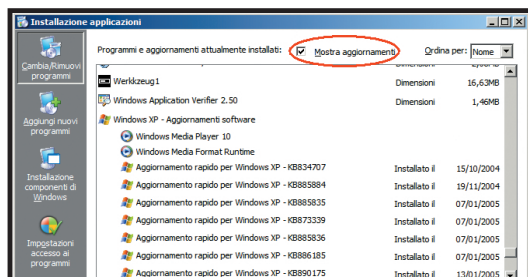



Fig. 1: ***Installazione applicazioni mostra solo una parte delle informazioni relative ad un aggiornamento***

Nelle versioni future verrà introdotto il supporto per la scansione di una rete, gli amministratori potranno così sviluppare script ed applicazioni con avanzati schemi di distribuzione delle patch.

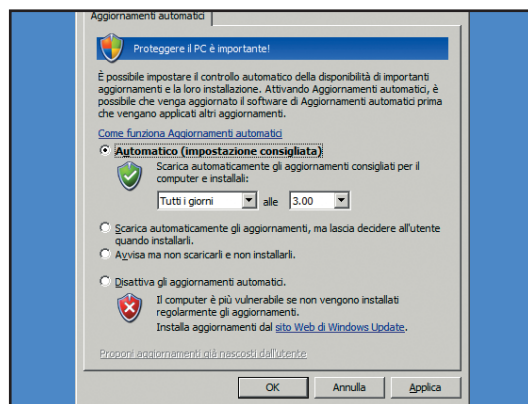



Fig. 2: ***Il Centro Sicurezza PC Windows è accessibile dal Pannello di Controllo***

Per il momento dobbiamo accontentarci dell'analisi limitata al computer locale. Vedremo come realizzare, con poche righe di codice, un programma simile ad *Aggiornamenti automatici*.

NOZIONI DI BASE

Lo spazio è tiranno, dunque non soffermeremo la nostra attenzione sulla gestione di server *Windows Update Services (WUS)* né tanto meno sulle discussioni riguardanti l'opportunità di affidarsi ad un sistema automatico di aggiornamento. Rimbocchiamoci le maniche e controlliamo che il servizio *"Aggiornamenti automatici"* (**Figura 3**) sia in esecuzione o comunque avviabile in modo manuale: l'infrastruttura WUA poggia su tale componente.

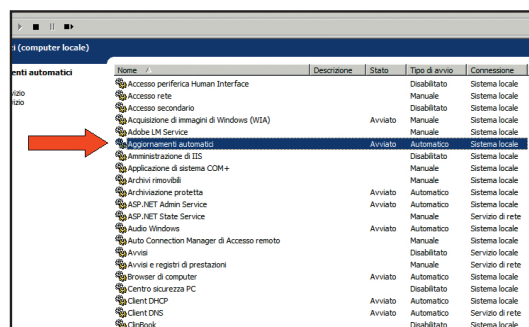



Fig. 3: ***Pannello di controllo -> Strumenti di amministrazione -> Servizi***

Il meccanismo viene messo in movimento dal motore *Windows Update AutoUpdate Engine* contenuto nella libreria a collegamento dinamico *\system32\wuaueng.dll*. WUA risulta presente quando la versione di *wuaueng.dll* è *5.4.3790.1000* o superiore (**Figura 4**). Svilupperemo l'applicazione di esempio utilizzando le interfacce esposte dalla libreria *\system32\wuapi.dll*, nota anche come *API*

REQUISITI

Conoscenze richieste
Basi di C#

Software
.Windows XP SP2 (o altri OS aggiornati) .NET Framework SDK

Impegno

Tempo di realizzazione

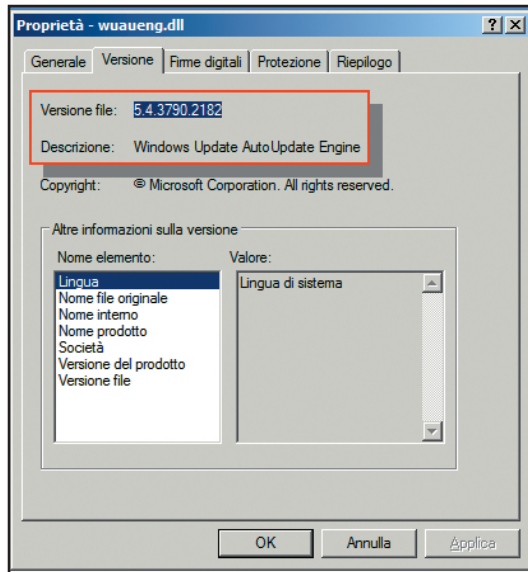



Fig. 4: ***WUA è installato se la versione di wuaueng.dll è >= 5.4.3790.1000***

Client di *Windows Update*. Per nostra fortuna il .NET Framework nasconde le complessità della programmazione COM. Sarà sorprendentemente semplice esaminare la lista degli aggiornamenti disponibili e procedere alla loro installazione dopo la fase di download. Poiché WUA fa uso del servizio *BITS (Background Intelligent Transfer Service)* i trasferimenti avvengono in modalità asincrona. Nel caso in cui si perda la connessione nel mezzo di un trasferimento, il processo di download continuerà al collegamento successivo. Sul fronte della sicurezza c'è da notare che i dati scambiati sono protetti dal protocollo *SSL (Secure Socket Layer)*, inoltre WUA installerà soltanto i pacchetti dotati di una firma digitale valida.

IOAGGIORNO

L'esempio proposto (**Figura 5**) è implementato in C# e richiede la presenza della libreria *WUApiLib .dll*. Tale file può essere generato aggiungendo un riferimento a *wuapi.dll* in Visual Studio .NET, Delphi 2005 o Sharp Develop, in alternativa si può ricorrere allo strumento *tlbimp.exe* incluso nel .NET Framework SDK. La procedura da seguire per ricreare da zero il progetto è illustrata nel riquadro *"Passi"*. Le interfacce impiegate da *IoAggiorno* sono essenzialmente quattro:

- **IUpdateSession** – ogni operazione di ricerca, download, installazione, disinstallazione, appartiene al contesto rappresentato da una sessione;
- **IUpdateSearcher** – i metodi più importanti sono *Search* e *QueryHistory*. Il primo effettua una ricerca basata sui criteri specificati, il secondo restituisce tutte le informazioni relative agli aggiornamenti installati sulla macchina.
- **IUpdateDownloader** – dopo aver indicato una collezione di aggiornamenti da scaricare si può passare al download.
- **IUpdateInstaller** – questa interfaccia permette di installare/disinstallare una collezione di aggiornamenti. Invocando il metodo *RunWizard* si crea un assistente in grado di guidare l'utente durante l'installazione.

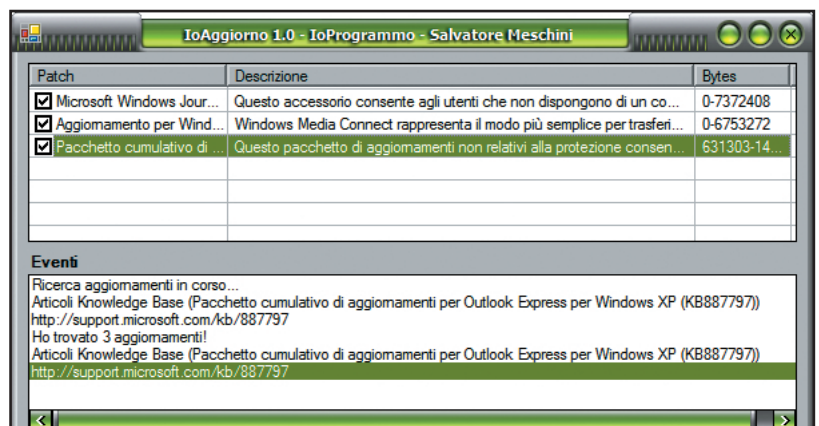
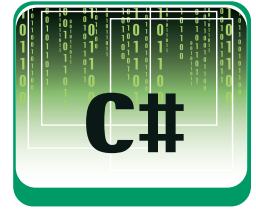



Fig. 5: ***L'applicazione di esempio IoAggiorno***

Le interfacce di WUA prevedono due diverse modalità: sincrona e asincrona. La prima è immediata da gestire ma risulta bloccante, la seconda è più complessa però le operazioni sono svolte in background. La scelta di una modalità rispetto all'altra è delegata al programmatore e dipende dalla tipologia di applicazione che si intende realizzare. *IoAggiorno* opera in modo sincrono e consente di: ricevere dal sito *Windows Update* la lista degli aggiornamenti disponibili, scaricare/installare in modo selettivo i pacchetti e visionare la sequenza degli aggiornamenti installati. Il programma, per aggiornare il sistema, richiede una connessione ad Internet precedentemente stabilita. La verifica dello stato di connessione è attuata dalla funzione *InternetGetConnectedState*

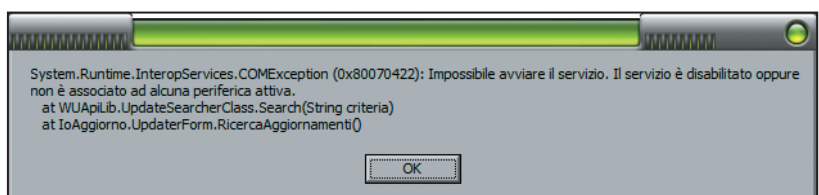
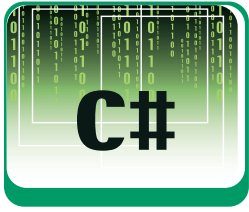



Fig. 6: ***Se il servizio "Aggiornamenti automatici" è disattivato viene visualizzato questo messaggio***

esportata da *wininet.dll.* Se il servizio *"Aggiornamenti automatici"* è disabilitato viene mostrato un messaggio di errore (**Figura 6**).
Il criterio di ricerca adottato da *IoAggiorno* è stabilito dalla chiamata al metodo *Search:*

```
Aggiornamenti = Ricerca.Search("IsInstalled=0 and
                                    Type='Software'");
```

In breve vengono rilevati gli aggiornamenti della tipologia *"Software"* non ancora installati *(IsInstalled =0).* Per cercare i driver più aggiornati basta sostituire *Type='Software'* con *Type='Driver'.* La documentazione di WUA illustra una decina di criteri impiegabili nelle ricerche.
Sul CD allegato alla rivista trovate il codice sorgente dettagliatamente commentato.

APPROFONDIMENTI

Per verificare la presenza di WUA si può ricorrere a *FileVersionInfo WuaVer = FileVersionInfo.Get VersionInfo(Environment.SystemDirectory+"\\wuaueng.dll");.* I campi *ProductMajorPart, ProductMinorPart, ProductBuildPart* e *ProductPrivatePart* di WuaVer devono rappresentare una versione maggiore o uguale alla 5.4.3790.1000. La classe FileVersionInfo appartiene allo spazio dei nomi *System.Diagnostics.*

IMPOSTAZIONI

Non abbiamo ancora visto come sia possibile accedere alle impostazioni di *Windows Update Agent.* Se vogliamo implementare un client WUA non possiamo prescindere da una simile caratteristica. Le impostazioni del servizio permettono di specificare il livello di notifica e la frequenza con cui ricercare gli aggiornamenti.

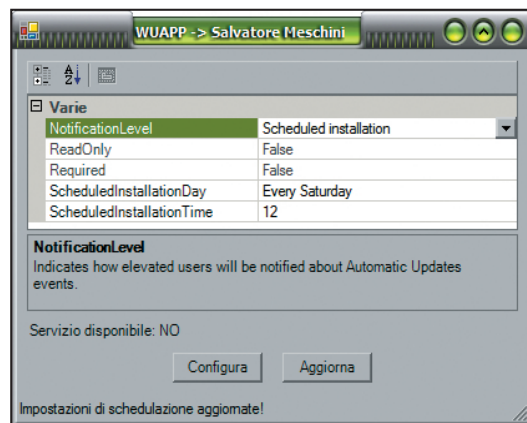



Fig. 7: ***PropertyGrid con le impostazioni di "Aggiornamenti Automatici"***

Il controllo *PropertyGrid* (**Figura 7**) è stato popolato dal codice seguente:

```
private void MostraConfigurazione()
{
  IAutomaticUpdates AU =  new AutomaticUpdates();
  IAutomaticUpdatesSettings Impostazioni = AU.Settings;
  // Visualizza le proprietà nella griglia
  ProprietàWUA.SelectedObject = Impostazioni;
  // Mostra lo stato di WUA
  EtichettaStato.Text += (AU.ServiceEnabled == true)
```

```
                                          ? "SI" : "NO";
}
```

L'interfaccia *IAutomaticUpdatesSettings* è dotata del metodo *Save(),* quindi, dopo aver modificato con banali assegnazioni la configurazione di *"Aggiornamenti automatici",* si può invocare:

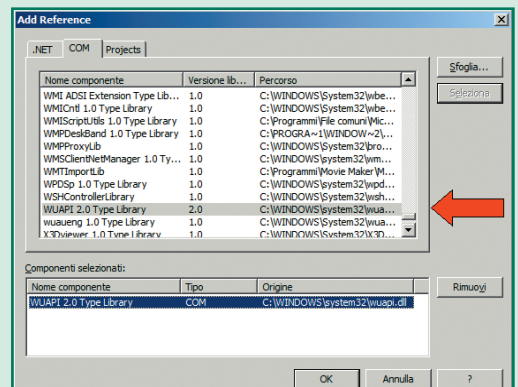
```
AU.Settings.Save();
```

Ovviamente non ha senso indicare il giorno della settimana e l'ora in cui aggiornare il sistema se la proprietà *NotificationLevel* è impostata ad un valore diverso da *"Scheduled Installation".* Per visualizzare la classica finestra di configurazione è sufficiente una sola istruzione:

```
AU.ShowSettingsDialog();
```

CREAZIONE DEL PROGETTO

1 **Assumendo che il vostro ambiente di sviluppo sia VS.NET è necessario creare un'applicazione Windows in C# *(CTRL+SHIFT+N).* Sharp Develop e Delphi 2005 prevedono progetti analoghi.**

RIFERIMENTO A WUAPI

2 **L'utilizzo delle interfacce WUA richiede un riferimento alla type library WUAPI (menu *Progetto->Aggiungi riferimento*). Si noti la presenza della clausola using *WUApiLib;* in *Form1.cs.***

Con poche righe di codice si ottiene la lista dettagliata degli aggiornamenti installati:

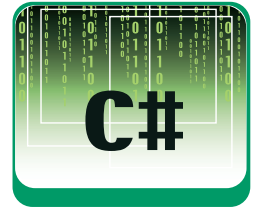
```
UpdateSession Sessione = new UpdateSession();
UpdateSearcher Ricerca =
                    Session.CreateUpdateSearcher();
// IUpdateSearcher ha il metodo GetTotalHistoryCount
IUpdateHistoryEntryCollection PatchPresenti =
              Ricerca.QueryHistory(0,Int32.MaxValue);
if (PatchPresenti.Count > 0)
{
  // Mostra i dettagli (titolo, data, etc.) delle patch
                                            rilevate
}
```

Le informazioni riguardanti il sistema si ricavano ricorrendo all'interfaccia *ISystemInformation.*
A titolo di esempio, se si vuole determinare la necessità di un riavvio al fine di completare un'installazione:

```
ISystemInformation info = new SystemInformation();
if (info.RebootRequired) MessageBox.Show(
                        "È necessario riavviare.");
```

CONCLUSIONI

L'applicazione IoAggiorno sfrutta solo una parte delle funzioni di WUA: la documentazione distribuita con il *Platform SDK*, reperibile anche tramite MSDN, descrive tutte le interfacce e le enumerazioni supportate.
A titolo di esempio è possibile utilizzare l'interfaccia *IDownloadProgress* per gestire una barra di progresso che mostri l'avanzamento delle operazioni di download. Lascio a voi il compito di esaminare i dettagli delle interfacce non descritte nell'articolo, vi garantisco che non mancheranno le sorprese. Alla prossima!

Salvatore Meschini

WUS
http://microsoft.com/wus
Gli amministratori di rete possono utilizzare un server WUS per distribuire ai sistemi client, in maniera centralizzata, gli aggiornamenti prelevati dal sito Windows Update.

MULTITHREADING

```
private void ListaBtn_Click(object sender, EventArgs e)
{
    Thread T = new Thread(new ThreadStart(RicercaAggiornamenti));
    // Avvia la ricerca di eventuali aggiornamenti (multithread)
    T.Start();
}

// Scarica gli aggiornamenti selezionati
private void ScaricaBtn_Click(object sender, EventArgs e)
{
    Thread T = new Thread(new ThreadStart(ScaricaAggiornamenti));
    T.Start();
}

// Connessione presente?
private bool connesso()
{
    InternetConnectionState flags = 0;
    return InternetGetConnectedState(ref flags, 0);
}
```

3 **Si è preferito, per non scomodare i metodi asincroni, creare nuovi thread in modo da non perdere il controllo dell'applicazione durante le lunghe operazioni di: ricerca, download ed installazione.**

RICERCA

```
if (!connesso()) MessageBox.Show("Connessione a Internet non rilevata!");
else
}
Sessione = new UpdateSession();
UpdateSearcher Ricerca = Session.CreateUpdateSearcher();
// Avvia la ricerca specificando un criterio:
Aggiornamenti = Ricerca.Search("IsInstalled=0 and Type='Software'");
for (int i=0; i <= Aggiornamenti.Updates.Count-1; i++)
{
ListaBox.Items.Add(Aggiornamenti.Updates[i].Title); // Nome della patch
ListaBox.Items[i].SubItems.Add(Aggiornamenti.Updates[i].Description);
...
}
}
```

4 **Nel metodo *RicercaAggiornamenti* viene creata una sessione di ricerca, le interfacce elencate nell'articolo sono dotate di una proprietà fondamentale: *Updates*. Sul CD trovate il codice commentato.**

PACCHETTI DA SCARICARE

```
UpdateCollection DaScaricare = new UpdateCollection();
  for (int i=0;  i <= Aggiornamenti.Updates.Count-1; i++)
  {
    // Gli aggiornamenti selezionati vengono aggiunti alla lista
    if (ListaBox.Items[i].Checked)
    {
      DaScaricare.Add(Aggiornamenti.Updates[i]);
    }
  }
UpdateDownloader Scaricatore = Session.CreateUpdateDownloader();
Scaricatore.Updates = DaScaricare; // Cosa vogliamo scaricare?
Scaricatore.Download(); // Procedi!
```

5 **Dopo aver aggiunto alla collezione *DaScaricare* i pacchetti da trasferire, è sufficiente creare un'istanza di *UpdateDownloader* con *CreateUpdateDownloader()* e richiamare il metodo *Download().***

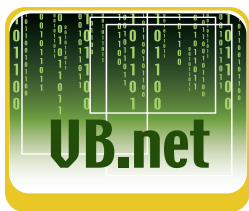
INSTALLAZIONE

```
UpdateCollection DaInstallare = new UpdateCollection();
  for (int i=0;  i <= Aggiornamenti.Updates.Count-1; i++)
  { if (ListaBox.Items[i].Checked)
    { if (Aggiornamenti.Updates[i].IsDownloaded)
      {
        DaInstallare.Add(Aggiornamenti.Updates[i]);
      }
      else
        Evento(Aggiornamenti.Updates[i].Title+" NON è stato
                                             scaricato." }
  UpdateInstaller Installatore = Session.CreateUpdateInstaller();
  Installatore.Updates = DaInstallare;
IInstallationResult Esito = Installatore.Install();
                                    // Avvia l'installazione
```

6 **Gli aggiornamenti scaricati tramite Windows *Update* vengono installati con il metodo *Install()* di *UpdateInstaller*. In alternativa si potrebbe gestire questa fase con chiamate a *BeginInstall* ed *EndInstall*.**

Scopriamo come cambiare l'aspetto dei nostri menu con VB.NET

Oggi cambiamo menu!

Impariamo come strutturare al massimo il framework .NET per realizzare menu estremamente personalizzati e con grafica accattivante

Utilizza questo spazio per le tue annotazioni

Chi utilizza un ambiente RAD come Visual Studio o simili, sarà ormai abituato a utilizzare questi strumenti in maniera visuale. Il metodo più semplice per disegnare l'interfaccia di un'applicazione è quello di prendere un elemento dalla casella degli strumenti e trascinarlo sulla form che rappresenta l'interfaccia. Utilizzando questi strumenti di disegno visuale delle applicazioni mi viene spesso da pensare che essi coprono almeno l'80% delle necessità per la progettazione di un'interfaccia utente, peccato che a noi sviluppatori, prima o poi, serve anche l'altro 20%! Questa considerazione è quanto mai vera per i menu, principali o contestuali, che sono un po' il "sale" delle nostre applicazioni. Un'interfaccia utente (UI per gli amici) ben realizzata contribuirà notevolmente alla facilità di utilizzo del vostro programma. Cosa c'è che non va nel buon vecchio menu che ci accompagna fin dalle prime edizioni del linguaggio Visual Basic? C'è che spesso risulta "drammaticamente" più spartano rispetto ai suoi "cugini" che gli utenti ritrovano nei programmi della suite di Office e simili. La prima differenza che salta agli occhi è la presenza, nei menu di Word & Company, di piccole icone accanto al testo come quelle in **Figura 1**. Lo scopo di questo articolo sarà quello di vedere come fare ad implementare dei menu con elementi grafici personalizzati in .NET.

Fig. 1: **Tipico menu di Office**

IL PROGETTO

Fig. 2: **Un oggetto MainMenu nell'area componenti dell'ambiente di sviluppo**

In Visual Studio, creiamo un *MainMenu* su una form. L'aggiunta del menu, di per sé, non produce niente di visibile nella nostra form. L'unico risultato "apprezzabile" è quello di veder comparire, nell'area componenti dell'ambiente di sviluppo, l'icona che vediamo in **Figura 2**. Se clicchiamo su questa icona, si attiva in Visual Studio l'Editor di menu che è poi quello che ci consente di aggiungere le varie voci, come possiamo vedere in **Figura 3**. Le varie voci che vengono aggiunte attraverso l'Editor di menu corrispondono ad altrettanti oggetti del tipo *System.Windows.Forms.MenuItem* che forniscono proprietà che consentono di configurare l'aspetto e la funzionalità della voce di

Fig. 3: **Editor di menu in Visual Studio**

menu. Tra le proprietà ce n'è una che, visto il nostro obiettivo, ci interessa particolarmente: *OwnerDraw .OwnerDraw* è una proprietà di tipo Boolean, di default impostata a *false* che indica se la voce di menu deve essere disegnata automaticamente dal sistema oppure se preferiamo farlo noi. Attratti dalla forza oscura della curiosità andremo ad impostare *OwnerDraw* su *true*, come in **Figura 4**, e proveremo a lanciare il programma con il consueto *F5* e che succede? Un disastro: al posto della voce di menu vediamo solamente un "rachitico" rettangolino. Beh… d'altra parte eravamo stati avvertiti: se la voce te la vuoi disegnare da te devi proprio disegnartela da solo, testo compreso! Bisogna rimboccarsi le maniche e scrivere qualche riga di codice utilizzando le classi grafiche di .NET.

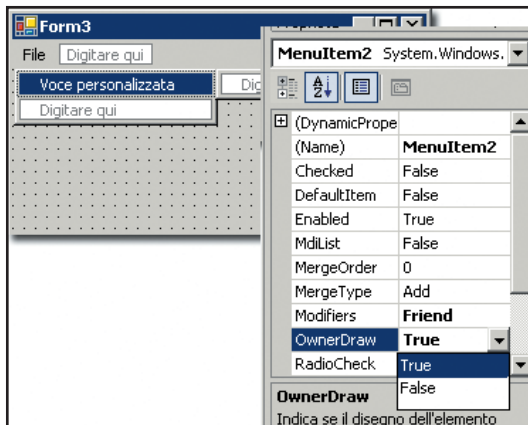



Fig. 4: **Impostazione della proprietà OwnerDraw**

PRENDIAMO LE MISURE

Mettetevi nei panni del programma. Quando vi si chiede "disegnami la voce di questo menu" la prima cosa che risponderete quale sarà? "Quali sono le dimensioni", naturalmente. Ed è proprio da qui che inizieremo. Ogni *MenuItem* ha una serie di *Eventi* associati tra cui appunto *MeasureItem* (misura l'oggetto) che serve appunto a conoscere la dimensione di una voce di menu prima di disegnarla. Lo schema del codice necessario a gestire l'evento è simile a

```
Private Sub MenuItem1_MeasureItem(ByVal sender
        As Object, ByVal e As System.Windows.Forms.-
                MeasureItemEventArgs) Handles
                        MenuItem1.MeasureItem
'aggiungere codice .....
End Sub
```

Tra i parametri notiamo: *"sender"* che è l'oggetto che ha originato l'evento (in questo caso il *MenuItem* da disegnare) ed *"e"* che è un'istanza della classe *System.Windows.Forms.MeasureItemEventArgs* che contiene il necessario per impostare la larghezza e l'altezza della voce di menu. Per impostare le dimensioni della voce del menu utilizziamo il seguente codice:

```
Private Sub MenuItem1_MeasureItem(ByVal sender As
    Object, ByVal e As System.Windows.Forms.Measure
      ItemEventArgs) Handles MenuItem1.MeasureItem
Dim myMenuItem As MenuItem = CType(sender, MenuItem)
    Dim altezza As Integer
    Dim larghezza As Integer
    Dim misuraDelTesto As SizeF
    misuraDelTesto = e.Graphics.MeasureString(
      myMenuItem.Text, SystemInformation.MenuFont)
    altezza = SystemInformation.MenuHeight
    larghezza = SystemInformation.MenuCheckSize
                .Width + misuraDelTesto.Width
    e.ItemHeight = altezza
    e.ItemWidth = larghezza
End Sub
```

In pratica troviamo le dimensioni del testo della voce del menu attraverso l'oggetto *Graphics* fornito da *System.Windows.Forms.MeasureItemEventArgs* chiamando la funzione *MeasureString* ed impostiamo larghezza ed altezza del menu attraverso *ItemWidth* e *ItemHeight*, proprietà sempre contenute in *System.Windows.Forms.MeasureItemEventArgs*. Si noti come, in questa fase, ci siamo attenuti alle dimensioni e font standard di sistema per i menu, recuperati attraverso la classe *SystemInformation*. Prima di personalizzare cerchiamo infatti almeno di riprodurre il comportamento di una normale voce di menu. Se lanciamo nuovamente la nostra applicazione vedremo adesso al posto della voce di menu un rettangolo vuoto, **Figura 5**. Ma tuttavia manca ancora il testo.

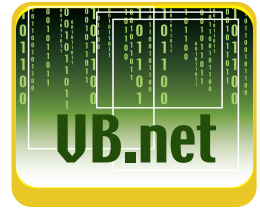
Fig. 5: **la voce di menu prende forma con MeasureItem**

DISEGNAMO

Torniamo nei panni del programma, dopo che gli sono state comunicate le dimensioni dell'area da disegnare di cosa avrà bisogno? Ovviamente di sapere cosa disegnare! È per questo che dobbiamo gestire un altro Evento: *DrawItem* (Disegna l'oggetto) con un metodo dal seguente schema:

```
Private Sub MenuItem1_DrawItem(ByVal sender
        As Object, ByVal e As System.Windows.Forms
    .DrawItemEventArgs) Handles MenuItem1.DrawItem
'aggiungere codice .....
End Sub
```

Anche qui nei parametri ritroviamo il buon vecchio *"sender"* (che è il nostro *MenuItem*) ed *"e"* che questa volta è un oggetto del tipo *System.Windows.Forms-*

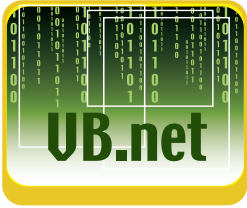
NOTA

.NET E LE CLASSI GRAFICHE

In .NET viene utilizzata un'implementazione avanzata dell'interfaccia di progettazione grafica (*GDI*) di Windows, denominata GDI+, che consente di creare grafica e testo, nonché modificare oggetti grafici come se fossero oggetti. GDI+ può essere utilizzato per eseguire il rendering di immagini grafiche in Windows Form e controlli. L'accesso alle classi grafiche avviene solitamente attraverso gli oggetti contenuti nel namespace *System.Drawing*.

TECNICA E LINGUAGGI

Negli esempi riportati nell'articolo e nel CD è stato utilizzato Visual Basic .NET ma le tecniche illustrate sono valide anche per gli altri linguaggi del Framework come C#.

IMAGELIST

Il componente ImageList fornisce i metodi per la gestione di un insieme di oggetti Image è possibile aggiungervi immagini bitmap, icone o metafile a ImageList; gli altri controlli possono utilizzare le immagini attraverso un riferimento all'indice della proprietà Items.

NOTA

LE CLASSI PER I MENU

L'oggetto menu corrisponde in .NET alle classi che discendono da *System.Windows.Forms.Menu*. Questa classe fornisce funzionalità comuni per tutte le classi di menu, in particolare System.Windows.Forms.MainMenu che rappresenta il menu principale dell'applicazione e System.Windows.Forms.ContextMenu che è il menu contestuale. Quello, per intenderci, che si attiva cliccando con il tasto destro su un controllo. MainMenu e ContextMenu sono composti da oggetti MenuItem. Questi possono contenere altri oggetti MenuItem, che rappresentano le voci dei sottomenu.

SUL WEB

Una buona risorsa per capire il funzionamento delle tecniche più avanzate è l'articolo "Owner-Drawing in .NET" pubblicato in MSDN Magazine al link:
http://msdn.microsoft.com/msdnmag/issues/04/02/CuttingEdge/default.aspx

.DrawItemEventArgs che ci metterà gentilmente a disposizione gli strumenti per disegnare. Pensiamo adesso alla struttura che dovrà avere la nostra voce di menu. Un rettangolo rappresenta l'intera area che contiene a sua volta due rettangoli: il più piccolo conterrà l'immagine, l'altro il testo.

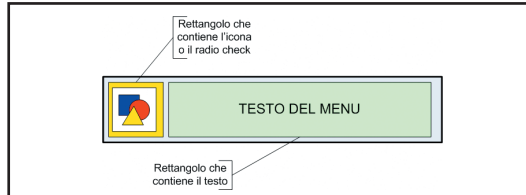



Fig. 6: *la voce di menu prende forma con MeasureItem*

L'intera area la conosciamo attraverso la proprietà Bounds (struttura di tipo *System.Drawing.Rectangle*) del *System .Windows.Forms.DrawItemEventArgs* che ci passa l'evento, i passaggi da fare sono quindi:

1. trovare i rettangoli dell'icona e del testo;
2. colorare con lo sfondo appropriato tutta la voce del menu;
3. se abbiamo un'immagine disegnarla nel rettangolo icona;
4. disegnare il testo nel rettangolo apposito.

L'intero metodo che gestisce l'evento *DrawItem* si presenterà quindi pressappoco così:

```
Private Sub MenuItem1_DrawItem(ByVal sender As
        Object, ByVal e As System.Windows.Forms
    .DrawItemEventArgs) Handles MenuItem1.DrawItem
   Dim ColoreTesto As Color = SystemColors.MenuText
   Dim ColoreSfondo As Color = SystemColors.Menu
   Dim RettangoloImmagine As System.Drawing
     .Rectangle = New Rectangle(e.Bounds.X, e.Bounds.
       Y, SystemInformation.SmallIconSize.Width,
         SystemInformation.SmallIconSize.Height)
   Dim AltezzaCarattere As Single
   AltezzaCarattere = e.Graphics.MeasureString(
                    sender.Text, Me.Font).Height
   Dim margineVerticale As Single = e.Bounds.Height -
                                  AltezzaCarattere
   Dim RettangoloTesto = New RectangleF(e.Bounds.X
      + RettangoloImmagine.Width, e.Bounds.Y +
       (margineVerticale / 2), e.Bounds.Width -
     RettangoloImmagine.Width, e.Bounds.Height -
                               margineVerticale)
   e.Graphics.FillRectangle(New SolidBrush(
                     ColoreSfondo), e.Bounds)
   e.Graphics.DrawString(sender.Text, Me.Font, New
       SolidBrush(ColoreTesto), RettangoloTesto)
 End Sub
```

Il metodo già consente di disegnare la voce di menu, ma noi vogliamo appunto aggiungere un immagine accanto al testo. Per semplicità, includiamo,nella finestra di progettazione, un componente ImageList che non è altro che un contenitore di immagini e, agendo sulla proprietà *Images* della finestra proprietà, vi inseriamo una piccola immagine in formato GIF della dimensione di una piccola icona (16x16 pixel). A questo punto, da codice, l'immagine potrà essere recuperata semplicemente con un riferimento all'indice:

```
Dim Immagine As Image = ImmaginiMenu.Images(0)
```

Oltre all'immagine dovremo però gestire anche lo "stato" della voce di menu, ovvero se il mouse vi è posizionato sopra o no. Per ottenere lo stato interroghiamo il risultato della congiunzione bit per bit della proprietà state di *DrawItemEventArgs* che è una combinazione dei membri dell'enumerazione dell'oggetto *DrawItemState:*

```
Dim selezionato As Boolean
selezionato = (e.State And DrawItemState.Selected) =
                        DrawItemState.Selected))
```

Il codice che gestisce *DrawItem* andrà quindi aggiornato con il disegno dell'immagine e la gestione dello sfondo e del testo in base allo stato, selezionato o meno:

```
Private Sub MenuItem2_DrawItem(ByVal sender As
        Object, ByVal e As System.Windows.Forms
   .DrawItemEventArgs) Handles MenuItem2.DrawItem
   Dim ColoreTesto As Color = SystemColors.MenuText
   Dim ColoreSfondo As Color = SystemColors.Menu
   Dim Immagine As Image = ImmaginiMenu.Images(0)
   Dim selezionato As Boolean = ((e.State And
     DrawItemState.Selected) = DrawItemState.Selected)
  If selezionato Then
    ColoreTesto = SystemColors.HighlightText
    ColoreSfondo = SystemColors.Highlight
  End If
  Dim RettangoloImmagine As System.Drawing
    .Rectangle = New Rectangle(e.Bounds.X, e.Bounds.
      Y, SystemInformation.SmallIconSize.Width,
        SystemInformation.SmallIconSize.Height)
  Dim AltezzaCarattere As Single
  AltezzaCarattere = e.Graphics.MeasureString(
                   sender.Text, Me.Font).Height
  Dim margineVerticale As Single = e.Bounds.Height -
                                 AltezzaCarattere
  Dim RettangoloTesto = New RectangleF(e.Bounds.X
 + RettangoloImmagine.Width, e.Bounds.Y + (margine
    Verticale / 2), e.Bounds.Width - Rettangolo
 Immagine.Width, e.Bounds.Height - margineVerticale)
  e.Graphics.FillRectangle(New SolidBrush(
                  ColoreSfondo), e.Bounds)
  e.Graphics.DrawImage(Immagine, RettangoloImmagine)
  e.Graphics.DrawString(sender.Text, Me.Font, New
```

```
                    SolidBrush(ColoreTesto), RettangoloTesto)
End Sub
```

Mandiamo in esecuzione il programma et voilà: il nostro duro lavoro sarà ricompensato da un menu con una voce decorata dalla piccola icona come quella che vediamo in **Figura 7**!

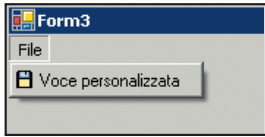
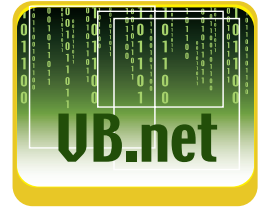
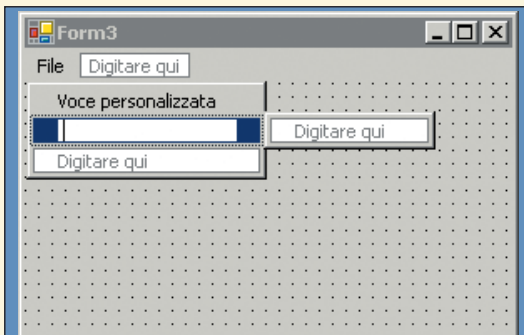



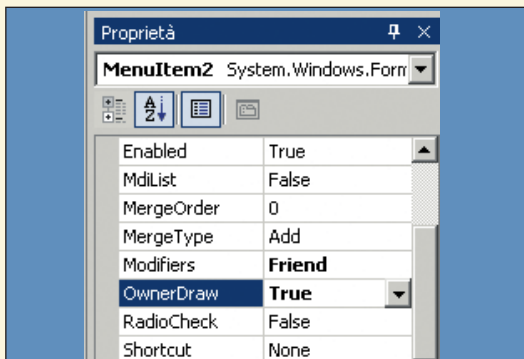
Fig. 7: ***il risultato finale***

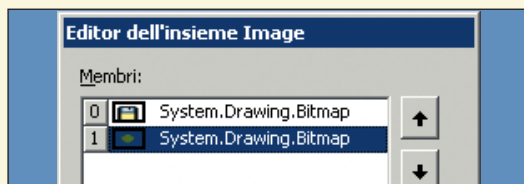
CONCLUSIONI

Abbiamo visto la tecnica che è alla base del disegno personalizzato dei Menu.
Tecnica che può anche essere utilizzata per la creazione di veri è propri componenti personalizzati ereditando dall'interfaccia *IExtenderProvider*, componenti magari dotati delle funzionalità di *design-time* all'interno di Visual Studio.

Francesco Smelzo

UN MENU CON ICONE IN SEI MOSSE!

DISEGNO DEL MENU

1 **In Visual Studio avviamo un progetto del tipo Window Application e su una Form trasciniamo dalla casella degli strumenti un oggetto MainMenu provvedendo a disegnare le varie voci contenute.**

PROPRIETÀ OWNERDRAW

2 **Selezionata la voce del menu (*MenuItem*) a che vogliamo personalizzare, nella finestra proprietà impostiamo la voce *OwnerDraw* sul valore *true*.**

CONTROLLO IMAGELIST

3 **Aggiungiamo alla form un componente ImageList nel quale inserire le immagini in formato 16x16 pixel da mostrare accanto alle voci di menu.**

GESTIAMO L'EVENTO MEASUREITEM

```
Private Sub MenuItem2_MeasureItem(ByVal sender As Object, ByVal e As
                    MeasureItemEventArgs) Handles MenuItem2.MeasureItem
  Dim myMenuItem As MenuItem = CType(sender, MenuItem)
  Dim altezza As Integer
  Dim larghezza As Integer
  Dim misuraDelTesto As SizeF
  misuraDelTesto = e.Graphics.MeasureString(myMenuItem.Text,
                                        SystemInformation.MenuFont)
  altezza = SystemInformation.MenuHeight
  larghezza = SystemInformation.MenuCheckSize.Width + misuraDelTesto.Width
  e.ItemHeight = altezza
  e.ItemWidth = larghezza
End Sub
```

4 **Definiamo un gestore dell'evento MeasureItem che imposta le dimensioni della voce di menu personalizzata.**

AREE DI DISEGNO,IMMAGINE E COLORI

```
Private Sub MenuItem2_DrawItem(ByVal sender As Object, ByVal e As
                          DrawItemEventArgs) Handles MenuItem2.DrawItem
  Dim ColoreTesto As Color = SystemColors.MenuText
  Dim ColoreSfondo As Color = SystemColors.Menu
  Dim Immagine As Image = ImmaginiMenu.Images(0)
  Dim RettangoloImmagine As System.Drawing.Rectangle=New Rectangle(e.Bounds.X,
          e.Bounds.Y, SystemInformation.SmallIconSize.Width, SystemInformation
                                                   .SmallIconSize .Height)
  Dim AltezzaCarattere As Single
  AltezzaCarattere = e.Graphics.MeasureString(sender.Text, Me.Font).Height
  Dim margineVerticale As Single = e.Bounds.Height - AltezzaCarattere
  Dim RettangoloTesto = New RectangleF(e.Bounds.X + RettangoloImmagine.Width,
     e.Bounds.Y + (margineVerticale /2), e.Bounds.Width - RettangoloImmagine.Width,
                                      e.Bounds.Height - margineVerticale)
(…)
End Sub
```

5 **Definiamo un gestore dell'evento DrawItem impostando le aree dove verranno disegnati immagine e testo ed i relativi attributi prendendo l'immagine dall'ImageList in precedenza creata.**

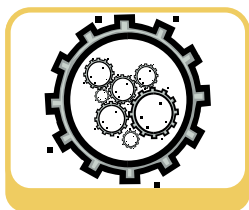
DISEGNO DEGLI ELEMENTI

```
Private Sub MenuItem2_DrawItem(ByVal sender As Object, ByVal e As
                          DrawItemEventArgs) Handles MenuItem2.DrawItem
  (…)
  e.Graphics.FillRectangle(New SolidBrush(ColoreSfondo), e.Bounds)
  e.Graphics.DrawImage(Immagine, RettangoloImmagine)
  e.Graphics.DrawString(sender.Text, Me.Font, New SolidBrush(ColoreTesto),
                                                         RettangoloTesto)
  End Sub
```

6 **Sempre all'interno del gestore dell'evento DrawItem disegniamo lo sfondo, l'immagine ed il testo.**

Dotare i propri software di un linguaggio interno

LUA: il linguaggio del software

Aumentare la flessibilità del nostro software, aggiungendo un vero e proprio interprete di comandi testuali, potrebbe sembrare un lavoro molto oneroso. In realtà con LUA tutto è più semplice

Utilizza questo spazio per le tue annotazioni

Molti dei software più complessi in circolazione non limitano l'interazione con l'utente alla sola interfaccia grafica. Chi lavora con programmi professionali, sa che è possibile estendere le varie funzionalità scrivendo un codice specifico per il software in questione. Questo codice, organizzato in "script", viene eseguito dal programma "ospite" e consente una vera e propria programmazione "ad alto livello".

SCRIPTING CON LUA

Uno script è su un semplice file di testo, né più né meno che il sorgente di un qualsiasi programma C o Java. Questo file può seguire due strade. Può essere elaborato così com'è da un programma detto *interprete*, oppure può essere trasformato in codice di basso livello, da un programma detto *compilatore*. Sia che si scelga la prima o la seconda strada, lo script passerà per una *Virtual Machine (VM)*, che si occupa di fornire l'ambiente per l'esecuzione vera e propria. Esistono diversi linguaggi di scripting, noi descriveremo in particolare LUA che è un pacchetto disponibile gratuitamente al link *www.lua.org*. Qui si possono trovare anche i file di installazione, la documentazione ufficiale, le librerie, i sorgenti e un libro di apprendimento on-line. Non manca nulla insomma!

Vedremo di seguito i fondamenti della sintassi di questo linguaggio. Chi è già avvezzo alla programmazione non dovrebbe trovarsi spaesato, anche se sono presenti diverse peculiarità alle quali prestare attenzione. Probabilmente la cosa più semplice da fare è cominciare con un esempio, infatti eccolo:

```
-- Il mio primo script con LUA
x = 23
y = "Ciao mondo!";
for z = 1, x do
print (y, " --> ", z);
end
```

L'ESEMPIO

Come è possibile intuire, l'effetto dello script precedente è quello di stampare a schermo per 23 volte la stringa "Ciao mondo!". Analizziamo riga per riga il codice. La prima riga è un commento e non influisce sulla funzionalità del codice ma solamente sulla sua leggibilità. Tutti i commenti in LUA sono su una singola riga e cominciano con "—". Non esiste quindi una sintassi particolare per i commenti su più righe, un po' scomodo ma… è così!

Nelle successive due righe vengono assegnati valori alle variabili *x* e *y*. Come si può vedere LUA è un linguaggio *"typeless"*, cioè non è previsto che si specifichi il tipo di ciascuna variabile (intero, stringa, carattere ecc.), al contrario di quanto avviene in altri linguaggi. È l'interprete, o il compilatore, a gestire nel modo che ritiene più ragionevole, le varie situazioni. Se ad esempio scriviamo:

```
a = 3;
b = 2;
print (a + b);
```

verrà correttamente stampato il risultato della somma: 5. Col codice:

```
c = "d vale: ";
d = 10;
print (c + d);
```

l'interprete segnalerà il seguente errore :

lua: prova.lua:3: attempt to perform arithmetic on

global `c' (a string value)

non si può infatti sommare una stringa a un intero, e non esiste una ragionevole conversione di tipi.
Per la cronaca, se quello che si potrebbe intendere col simbolo "+" è il concetto di "concatenazione di stringa e intero", è possibile ottenere questo risultato usando la virgola "," come infatti avviene poche righe dopo. Sempre a proposito di punteggiatura, segnaliamo che la presenza del terminatore di istruzione ";" è opzionale. Ogni riga potrebbe essere terminata semplicemente dal carattere di "a capo", tuttavia LUA cercherà di interpretare come istruzioni multi-linea le righe che probabilmente presentano un errore o sono incomplete. Insomma, meglio mettere il ";" che non metterlo: si evitano confusioni.
Le ultime tre righe dell'esempio mostrano un semplice ciclo, realizzato col costrutto "for". L'istruzione tra il for e la clausola conclusiva "end" viene ripetuta assegnando alla variabile in oggetto (z) i valori dell'intervallo specificato (da 1 a x, cioè da 1 a 23).
È da notare come il limite superiore (23) sia effettivamente un valore assegnato. Questo può generare un po' di confusione da chi viene da altri linguaggi, per cui è necessario prestare la dovuta attenzione per evitare errori difficili da scovare in fase di debug.
Questa caratteristica fa il paio con un'altra peculiarità di LUA: quando si dichiara un array di valori (in realtà una tabella, come vedremo) l'indice del primo elemento non è 0, bensì 1.
LUA supporta altri due tipi di iterazione, che faranno sentire a proprio agio i programmatori delle più varie estrazioni. È infatti possibile utilizzare sia il costrutto *"do-while"* che *"repeat-until"*:

```
local i = 1;
while i ~= 5 do
        print ("Ciclo while! --> ", i);
        i = i+1;
end
i = 1;
repeat
print ("Ciclo repeat! --> ", i);
        i = i+1;
until i == 5
```

I due cicli eseguono per 4 volte ciascuno (attenzione!) una stampa a schermo, incrementando la variabile i. Da notare come il simbolo per il test di disuguaglianza (~=) sia leggermente differente da quello che si po' trovare in altri linguaggi. Le condizioni per ottenere il medesimo risultato nel while e nel repeat sono inoltre opposte tra loro, come ci si deve aspettare. Un'altra cosa degna di nota è la dichiarazione della variabile i con la clausola "local".
In LUA tutte le variabili hanno una visibilità globale una volta istanziate. Per ottenere un comportamento differente da questo, ad esempio se si vuole dichiarare una variabile localmente a una funzione, si usa questa parola chiave.

CONTROLLO DI FLUSSO

Le condizioni logiche sono ovviamente usate anche dall'altra istruzione che si occupa del controllo del flusso del programma: il costrutto *"if-then-else"*:

```
-- if semplice
if a < 0 then
        a = 0;
end
-- clausola "else"
if b == a then
        b = b+1;
else
        b = b-1;
end
```

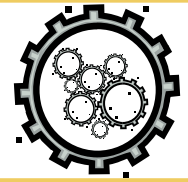
La sintassi è abbastanza intuitiva. Non è presente in LUA un costrutto simile allo "switch" del C/C++ ma si può ovviare a questa cosa utilizzando la clausola *"elseif"* che consente di concatenare una serie di if in sequenza, omettendo la clausola di chiusura *"end"*:

```
if a > 0 then
        print ("a è positivo!");
elseif a < 0 then
        print ("a è negativo!");
else
        print ("a == 0!");
```

NIL

Sia i costrutti di ciclo che l'*if* prevedono un test su una condizione. L'esito di questo test può essere vero *(true)* o falso *(false)*. L'unico concetto di falso che esiste in LUA è il *nil*. *Nil* è un valore "speciale", differente da qualsiasi altro valore assegnabile o tipo utilizzabile. Serve in LUA a rappresentare l'assenza di un valore utile per una variabile. Ogni variabile non inizializzata assume il valore nil per default. È possibile inoltre assegnare a una variabile globale il valore *nil* per cancellarla.
Facciamo un esempio, illustrando un altro concetto, quello di multi-assegnazione. È possibile scrivere in LUA una cosa del genere:

x, y, z = 1, 2, 3;

Questa istruzione assegnerà, rispettivamente a x, y e z i valori 1, 2 e 3. È quindi perfettamente equivalente a:

UTILIZZARE LUA SOTTO WINDOWS

LUA si presenta come un pacchetto software molto snello, ottenibile scaricando dal sito ufficiale www.lua.org il file lua-5.0.2.tar.gz. LUA prevede sia un interprete interattivo che un compilatore. I nomi sono "*lua.exe*" e "*luac .exe*". Non sono tuttavia presenti nel pacchetto base in forma binaria. Per questo conviene scaricarli dal link http://lua-users.org/wiki/LuaBinaries. Per utilizzare l'interprete è sufficiente aprire un prompt dei comandi e portarsi nella cartella contenente il file *lua.exe* ed eseguirlo. Un esempio di interazione è il seguente:

```
> a = 23 [INVIO]
> print (a) [INVIO]
23
>
```

Per eseguire lo script di esempio che si trova sul CD, è sufficiente digitare:

```
lua lua_parte1.lua
```

Per compilare lo script di prova bisogna utilizzare luac.exe:

```
luac -o lua_parte1.out
        lua_parte1.lua
```

NOTA

PERCHÉ ESISTE LUA

Questo linguaggio è stato creato per rendere semplice la soluzione di problemi utilizzando poche righe di codice. I suoi punti di forza sono:

- **ESTENDIBILITA: LUA è progettato per essere integrato con altri linguaggi come C/C++, Java, Fortran ecc.**
- **SEMPLICITÀ: LUA è un linguaggio snello e semplice, facile da apprendere ma al tempo stesso molto potente.**
- **EFFICIENZA: per le caratteristiche che offre, LUA presenta una implementazione alquanto efficiente.**
- **PORTABILITÀ: LUA è disponibile per tantissime piattaforme, non solo Windows o Unix/Linux.**

x = 1;
y = 2;
z = 3;

Cosa succede se il numero degli elementi di destra e sinistra non corrisponde? Una cosa molto intuitiva. Nel caso di valori in soprannumero, come ad esempio:

x, y = 1, 2, 3;

il valore in eccesso (il 3) verrà semplicemente scartato, scomparendo senza lasciare traccia. Nel caso di variabili in eccesso invece, come:

x, y, z = 1, 2;

l'elemento non valorizzato sarà automaticamente posto a nil:

```
a, b = "Ciao!";
if b then
        print( "Non verrò mai eseguita! :(");
else
        print( "E' ovvio che tu sia qui!");
end
```

FUNZIONI

LUA prevede (ovviamente!) la possibilità di definire funzioni. Le funzioni possono essere utilizzate per eseguire un compito specifico (subroutine), oppure all'interno di espressioni per calcolare un valore. La lista degli argomenti va racchiusa tra parentesi tonde e, se non ci sono argomenti, va indicata la lista vuota: (). Esistono particolari casi in cui le parentesi possono essere omesse, ma noi omettiamo di riportarli! Da una parte infatti è sempre opportuno mantenere uno stile omogeneo di scrittura, dall'altra il lettore interessato avrà uno stimolo in più per leggere la documentazione ufficiale...
Per quello che riguarda invece la definizione di una funzione la sintassi è abbastanza convenzionale:

```
-- Utilizzo come calcolo di un valore
function Seno (arg)
-- funzione della libreria "math"
        return math.sin(arg);
end
-- Utilizzo come subroutine
function Saluta(quantevolte)
        -- local i = 1;
        if quantevolte > 0 then
                for i = 1, quantevolte do
                        print ("Ciao!");
                end
        else
                print ("Ciao!");
        end
end
```

I parametri sono trattati esattamente come variabili locali e inizializzati col valore passato nella chiamata. È possibile passare un numero di parametri diverso da quello indicato nella lista. Il comportamento di LUA sarà simile a quello già visto nel caso di multi-assegnazione. Nel caso di parametri in eccesso i valori superflui verranno scartati. Nel caso invece di parametri in meno, quelli senza inizializzazione verranno posti a nil. Questo comportamento potrebbe generare problemi se trascurato, tuttavia può rivelarsi utile qualora i parametri non passati debbano assumere dei valori di default. Basta infatti fare un semplice test all'interno della funzione sui valori che sono nil, per capire se hanno bisogno di essere inizializzati o meno:

```
function valore_default (a)
  if a == nil then
    print ("Non e' stato passato alcun valore!");
  else
    print ("Il parametro vale: ", a);
  end
end
valore_default();
valore_default(23);
```

TABELLE

Le tabelle non sono una struttura dati di LUA, ma la struttura dati per eccellenza. Tutti i tipi complessi vengono realizzati in LUA tramite tabelle, non da ultime le strutture dati che consentono di utilizzare il paradigma di programmazione "object oriented". Le tabelle di LUA sono degli "insieme associativi", cioè strutture dati che fanno corrispondere a un indice (o "chiave") un determinato valore. Le tabelle possono essere utilizzate alla stregua dei normali array C/C++:

```
ArrayDiInteri = { 23, 46, 100};
ArrayDiStringhe = { "Ciao ", "Mondo!" };
```

È possibile accedere agli elementi di una tabella utilizzando le parentesi quadre []. L'unica accortezza da adoperare è, come già accennato, ricordarsi che l'indice di partenza non è 0 ma 1:

```
-- Stampa "Ciao Mondo!"
print (ArrayDiStringhe[1], ArrayDiStringhe[2]};
```

Nonostante l'inizializzazione di N elementi imposti un range di indici da 1 a N, è sempre possibile assegnare un valore a un qualsiasi indice al di fuori di

questo intervallo:

```
ArrayDiStringhe[23] = "Ciao Marte!";
print (ArrayDiStringhe[23]);
```

Come è facile indovinare, provando ad accedere a una tabella con un indice non inizializzato, ci ritroveremo a fare i conti col famigerato nil:

```
print (ArrayDiStringhe[17]);
```

stamperà: *nil*. Un aspetto fondamentale, che conferisce notevole flessibilità a questa struttura dati, è il fatto che le tabelle possono essere **eterogenee**. In altre parole una tabella può contenere valori di diversi tipi:

```
ArrayGenerico = {}; -- Un modo di dichiarare una tabella
ArrayGenerico[1] = 23;
ArrayGenerico[2] = "Ri-ciao Mondo!";
```

Questo vuol dire che un elemento di una tabella può essere a sua volta una tabella:

```
ArrayGenerico[3] = { 46, "Basta coi saluti, Mondo!"};
print (ArrayGenerico[3][2]); -- Doppio indice per l'accesso
-- alla tabella interna
```

È importante a questo punto sottolineare come gli elementi di una tabella siano tutti dei **riferimenti** ai valori originali e non delle copie degli stessi. Il seguente codice:

```
stringa = ArrayGenerico;
stringa[3][2] = "Magia!";
print (ArrayGenerico[3][2]);
```

stamperà: *Magia!* Nonostante non sia stata modificata direttamente la tabella ArrayGenerico. In altre parole nelle tabelle vengono conservati i "puntatori" ai valori immessi e non i valori medesimi. Il concetto di "array associativo" è ancora di più accentuato dalla possibilità di considerare come indici delle vere e proprie "chiavi", rappresentate da stringhe. Analogamente a quanto avviene per le hashtable presenti in altri linguaggi, in LUA è possibile utilizzare le tabelle per memorizzare delle coppie "chiave-valore":

```
TabellaChiavi["Nome"] = "Alfredo";
TabellaChiavi["Cognome"] = "Marroccelli";
TabellaChiavi["Eta"] = 26;
```

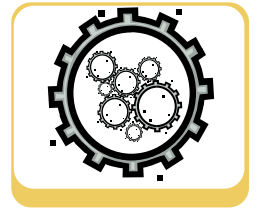
Questo dovrebbe rendere evidente la possibilità di utilizzare le tabelle come strutture dati con campi personalizzabili. La cosa è ancor più vera se si considera uno dei modi possibili di fare riferimento a questi valori, utilizzando la sintassi col punto ".":

```
print ("Ciao! Sono ", TabellaChiavi.Nome,
                              TabellaChiavi.Cognome,
"\n e ho ", TabellaChiavi.Eta, " anni!");
```

L'utilizzo profondo delle enormi potenzialità offerte da questa struttura dati è una cosa che richiede molta esperienza. Molti dei "trucchi" utilizzabili, infatti, sono difficili da cogliere per un neofita, ma conferiscono agli script in LUA un aspetto del tutto particolare! Consigliamo quindi molta pratica in questo campo, unita alla lettura di codice scritto da programmatori esperti, reperibile in quantità nella rete.

CONCLUSIONI

In questo articolo sono state trattate le basi della programmazione di script in LUA. Abbiamo visto come questo linguaggio abbia sia punti di contatto con i tradizionali linguaggi di programmazione, sia interessanti potenzialità aggiuntive. Come l'essere typeless o la straordinaria flessibilità delle tabelle, solo per citarne alcune. Il grosso delle potenzialità di LUA, tuttavia, risiede nella sua possibilità di "embedding" in un software "ospite". Sarà questo l'argomento del prossimo articolo, non mancate!

Alfredo Marroccelli

UNA COMUNITÀ MONDIALE!

I progetti che includono delle funzionalità implementate con LUA sono davvero tantissimi. Per questo motivo la comunità di sviluppatori/utilizzatori in tutto il mondo ha delle dimensioni non trascurabili. I link di riferimento dei "portali" su LUA sono http://lua-users.org/ e http://lua-forge.net/. In particolare cliccando su http://lua-users.org/wiki/LuaAddons si ha accesso a una serie di tools e add-on impressionante.

GETTING STARTED

STRINGHE

```
Il seguente codice stampa una stringa
-- e la sua lunghezza
s = "Ciao!";
print (s);
print (string.len(s));
```

1 **LUA prevede una serie di librerie aggiuntive. Una parte delle funzioni fornite sono per la manipolazione di stringhe.**

SOTTOSTRINGHE

```
s2 = "ioProgrammo con LUA!";
-- Specifico stringa originale e estremi,
                              partendo da 1
s3 = string.sub(s2,6,12);
-- L'estrazione di una stringa non
                         modifica l'originale
print (s2); --> "ioProgrammo con LUA!"
print (s3); --> "grammo"
```

3 **È molto semplice estrarre una parte di stringa da una più grande, specificando gli estremi. Come al solito l'originale non viene modificata.**

MAIUSCOLE E MINUSCOLE

```
Modifico i caratteri della stringa
                              precedente
s_MAIUSC = string.upper(s);
s_minusc = string.lower(s);
-- Stampa "CIAO!" e "ciao!"
-- Il carattere "!" non viene modificato
print (s_MAIUSC);
print (s_minusc);
```

2 **È possibile modificare le lettere maiuscole in minuscole e viceversa. La stringa modificata è fornita come risultato.**

FORMATTAZIONE

```
-- Stampa di un numero con 2 cifre
                                decimali
n = 12.3456;
print(string.format("n = %.2f", n)); -->
        "n = 12.35" (arrotonda...)
-- Stampa di una data con lunghezza
                          fissa dei campi
g = 17; m = 6; a = 2001;
print(string.format("%02d/%02d/%04d",
        g, m, a)); --> "17/06/2001"
```

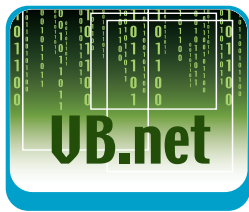
4 **Si può formattare la stampa di un numero analogamente a quanto avviene in C con la funzione *printf()*.**

Sfruttiamo il codice di Microsoft per Facilitarci la vita

Meno codice per tutti!

Vedremo come utilizzare le classi Data della libreria Application Blocks fornita da Microsoft per scrivere meno codice per accedere al database SQL Server

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

Le classi che compongono il .NET Framework formano già un buon livello di astrazione del codice, permettendo di scrivere applicazioni con minor numero di righe di codice rispetto ad altre tecnologie. A volte, però, le classi del .NET Framework non sono ottimizzate e il programmatore deve scrivere sempre le stesse righe di codice per ottenere una determinata funzionalità. Per fare un esempio concreto prendiamo la funzionalità di richiesta dati ad un database, utilizzando un DataSet come contenitore. I passi sono sempre gli stessi:

1. Creazione di un oggetto connessione tipo *SqlConnection* per specificare i parametri per collegarsi al database.
2. Definizione di un comando SQL per richiedere dati ad un database.
3. Creazione di un oggetto *DataAdapter* che utilizzi la connessione e il comando SQL specificati precedentemente per ricavare i dati.
4. Utilizzo del metodo *Fill()* del DataAdapter per inserire i dati ricavati dal database all'interno del DataSet.

Questi passi si traducono nel seguente codice:

```
SqlConnection conn = new
SqlConnection("Server=.;Database=Northwind;
                    Integrated Security=true");
SqlDataAdapter da = new SqlDataAdapter("SELECT *
                    FROM Products",conn);
DataSet ds = new DataSet();
da.Fill(ds);
```

Nulla di trascendentale, vero, ma immaginiamo che la nostra applicazione colloqui spesso con il database; il nostro codice sarebbe pieno di chiamate repetitive. Non sarebbe più semplice avere una situazione così?

```
DataSet ds = SqlHelper.ExecuteDataset(
ConnectionString, Data.CommandType.Text, "SELECT *
                    FROM Products");
```

Daltronde la nostra richiesta era semplice; in base a questa stringa di connessione riempimi il DataSet con i record ricavati dal comando SQL che ti fornisco. Quello che esegue il metodo *ExecuteDataset* contenuto all'interno della classe SqlHelper fornita dai *Microsoft Application Blocks* è proprio quello che ci serviva. Niente codice ripetitivo, un'istruzione diretta e facilmente manutenibile.

UTILIZZIAMO IL DATA APPLICATION BLOCK

Anche se abbiamo già visto come un metodo del Data application block permette di ottenere gli stessi risultati scrivendo meno codice, questa libreria fornisce altri metodi statici per effettuare altre operazioni interessanti come aggiornare i dati di una tabella tramite una transazione, ricavare dei dati utilizzando delle stored procedures, ecc. In questo articolo utilizzeremo il database NorthWind di MS SQL Server come supporto all'analisi del codice. Questa scelta è dovuta al fatto che il *Quick Start Example* che utilizzeremo per comprendere il funzionamento dell'Application Block sfrutta proprio il database di Microsoft. Nonostante questo l'Application Block supporta anche MSDE con le stesse modalità descritte in questo articolo. La differenza con i nostri

esempi risiede nel fatto che MSDE non supporta le stored procedure.

L'AMBIENTE

La prima operazione per utilizzare gli esempi forniti con l'installazione del Data Application Block è quella di eseguire uno script SQL all'interno del Query Analyzer in modo da aggiungere tabelle e stored procedures sulle quali lavorare. Eseguiamo il tool scegliendo la voce Query Analyzer dal menu Microsoft Sql Server presente tra i programmi di Windows e inseriamo le giuste credenziali per collegarci al database. Copiamo il contenuto dello script CreateStoredProcedures.sql presente nel progetto di Visual Studio .NET all'interno del Query Analyzer ed eseguiamo i comandi.

Il progetto *Quick Start Example* si trova nella directory di installazione dell'Application Block, è necessario aprirlo e compilarlo per avere un riferimento di quanto proposto nei paragrafi seguenti.

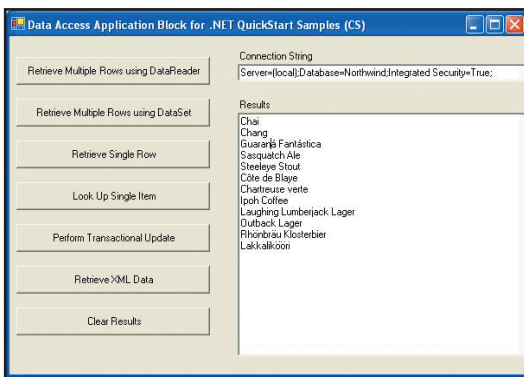



Fig. 1: *Il Quick Start Example in azione*

RICAVARE RECORD UTILIZZANDO IL DATAREADER

Nel primo esempio viene utilizzato un SqlDataReader per scorrere tra i record ricavati dall'esecuzione di una stored procedure.

Vediamo il codice utilizzato:

```
private void cmdSample1_Click(object sender,
                               System.EventArgs e)
{
   SqlDataReader dr;
   dr = SqlHelper.ExecuteReader(txtConnectionString.Text,
                               "getProductsByCategory", 1);
   txtResults.Clear();
   while (dr.Read())
   {
      txtResults.Text = txtResults.Text + dr.GetValue(1)
                           + Environment.NewLine;
   }
}
```

Tramite l'utilizzo del metodo statico *ExecuteReader* viene eseguita la stored procedure *getProductsByCategory* alla quale viene passato un parametro del valore 1 utilizzando la stringa di connessione specificata nella casella di testo della form principale. Tutto qui? Esatto, questo è tutto. Qualcuno potrebbe chiedersi chi si occupa di chiudere la connessione con il database dopo che il *DataReader* è stato utilizzato all'interno del ciclo *while*. La risposta la possiamo trovare all'interno del codice del *Data application block:*

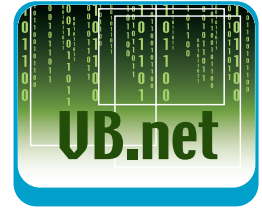
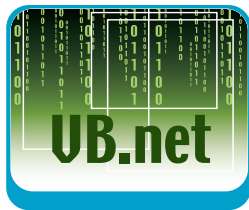
```
if (connectionOwnership ==
                  SqlConnectionOwnership.External)
{
   dr = cmd.ExecuteReader();
}
else
{
   dr = cmd.ExecuteReader(
             CommandBehavior.CloseConnection);
}
```

Qui il codice si chiede se la connessione è stata aperta dall'esterno, ovvero il programmatore si è preoccupato di gestire la connessione, oppure la gestisce il *Data application block*. Nel secondo caso il codice specifica di chiudere la connessione con il database alla chiusura del *DataAdapter* restituito dal metodo *ExecuteReader*.

RICAVARE RECORD UTILIZZANDO UN DATASET

Il codice contenuto all'interno del secondo pulsante permette di ricavare un DataSet riempito con i record ricavati dall'esecuzione della stored procedure *getProductsByCategory*:

```
DataSet ds;
ds = SqlHelper.ExecuteDataset(txtConnectionString.Text,
        CommandType.StoredProcedure, "getProducts
   ByCategory", new SqlParameter("@CategoryID", 1));
```

Il metodo statico *ExecuteDataset* accetta una stringa di connessione, un *CommandType* per specificare il tipo di commando fornito come terzo parametro e una lista di parametri, nel caso in cui il comando è una stored procedure. Il tutto è abbastanza semplice.

Il progetto è realizzato con la versione 2002 di Visual Studio .NET percui dovrete procedere alla conversione della soluzione nel caso lo utilizziate con una versione successiva dell'ambiente di sviluppo.

RICAVARE UN SINGOLO VALORE

Utilizzando la libreria ADO.NET è possibile ricavare un singolo valore come un aggregato (somma, valore massimo, ecc.) utilizzando la ExecuteScalar fornita dalla classe Command. Ma come al solito le operazioni da effettuare prima di chiamare questo metodo sono molte. Vediamo il codice dietro al pulsante Look up single item che utilizza una versione rivisitata del metodo ExecuteScalar:

```
productName = (string)SqlHelper.ExecuteScalar(
                    txtConnectionString.Text, CommandType.
                    StoredProcedure, "getProductName", new
                              SqlParameter("@ProductID", 1));
txtResults.Text = productName;
```

Il metodo viene utilizzato per ricavare il nome di un prodotto il cui identificativo è uguale al valore specificato tramite il parametro *@ProductID* della stored procedure *getProductName.*

INSERIMENTO DATI CON SUPPORTO DELLA TRANSAZIONE

Il metodo *ExecuteNonQuery* della classe *Command* permette di eseguire un comando SQL che non ritorni nessun valore, solitamente un comando di inserimento o modifica dei record di una tabella. Ad esempio, per utilizzare una stored procedure che inserisca dei dati è buona regola utilizzare una transazione in modo che se qualcosa non va per il verso giusto, questa possa essere utilizzata per riporta-

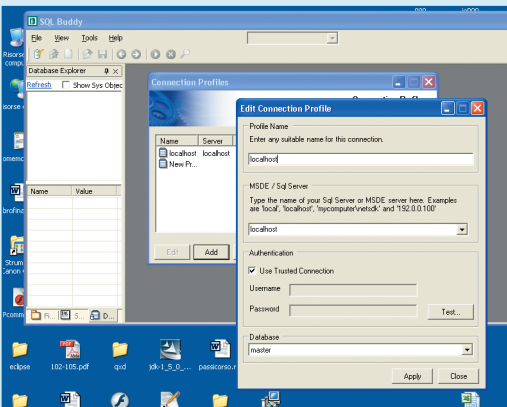
NOTA

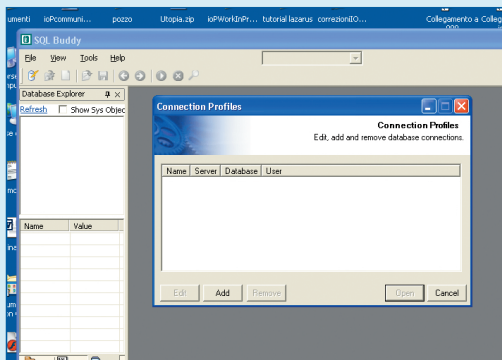
I MICROSOFT APPLICATION BLOCKS

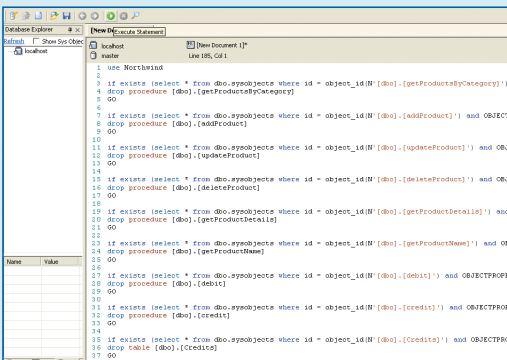
Nati come codice di supporto alla scrittura di applicazioni in C# e Visual Basic .NET i Microsoft Application Blocks sono degli assembly il cui codice è gratuitamente scaricabile dal sito http://download.microsoft.com. Sono presenti diversi Application Blocks; uno per la gestione del database, uno per la gestione della cache, uno per la gestione delle eccezioni, ecc. Una volta completata l'installazione sul proprio sistema del Data Application Block sarà possibile accedere al sorgente della libreria dal menu *Start* di Windows che presenta il nuovo gruppo Microsoft Application Blocks. All'interno di esso sono presenti anche dei collegamenti alla documentazione e ad esempi realizzati in C# e Visual Basic .NET.

CREARE QUERY IN MSDE/SQL SERVER

In questo articolo abbiamo utilizzato come base per gli esempi Ms Sql Serve di Microsoft. Il quick start fornito con l'application block fa uso infatti del database di NorthWind distribuito con MS Sql Server. Nonostante questo è ovviamente possibile utilizzare in modo efficiente l'application block anche con altri sistemi meno costosi. Le analisi fatte nell'articolo rimangono tutte valide anche per altri sistemi, anche se il quick start potrà essere eseguito solo basandosi su MS SQL Server. In ogni caso allegato alla rivista trovate SQL Buddy un ottimo query editor che vi consente di connettervi ad altri database, primo fra tutti MSDE. Di seguito illustriamo una breve procedura per utilizzare SQL Buddy in alternativa al Query Analyzer che normalmente non é disponibile con MSDE. Se utilizzate SQL Buddy per eseguire l'esempio proposto in questo articolo, dovrete eseguire la query contenuta nel file SetupDataBase.sql contenuto nella directory di installazione di SQL Buddy.

INSERIAMO I DATI

2 **È necessario indicare un nome per il profilo, il nome della macchina che esegue MSDE o MS SQL e l'eventuale login e password per l'autenticazione**

CREIAMO UN NUOVO PROFILO

1 **SQL Buddy è in grado di gestire piu di un database per volta. Ad ogni database deve essere associato un profilo contenente i dati per la connessione**

ESEGUIAMO LA QUERY

3 **Le query possono essere editate direttamente dalla finestra principale di SQL Buddy e mandate in esecuzione tramite il pulsante nella barra dei menu**

re la situazione dei dati nel momento esatto in cui la transazione è iniziata. Utilizzare una transazione in ADO.NET non è molto complesso ma richiede diversi accorgimenti che vengono facilitati dall'utilizzo della versione statica di ExecuteNonQuery fornita dalla libreria Data Application Block. Vediamo il codice utilizzato nell'esempio per utilizzare una transazione:

```
private void cmdSample5_Click(object sender,
                                    System.EventArgs e)
{
  using (SqlConnection conn = new
  SqlConnection(txtConnectionString.Text) )
  {
    conn.Open();
    using (SqlTransaction trans =
                              conn.BeginTransaction())
    {
      SqlParameter paramFromAcc = new
        SqlParameter("@AccountNo", SqlDbType.Char,
                                                  20);
      paramFromAcc.Value = "12345";
      SqlParameter paramToAcc = new SqlParameter(
              "@AccountNo", SqlDbType.Char, 20);
      paramToAcc.Value = "67890";
      SqlParameter paramCreditAmount = new
        SqlParameter("@Amount", SqlDbType.Money );
     paramCreditAmount.Value = 500;
     SqlParameter paramDebitAmount = new
       SqlParameter("@Amount", SqlDbType.Money );
     paramDebitAmount.Value = 500;
     try
     {
       SqlHelper.ExecuteNonQuery(trans,
             CommandType.StoredProcedure, "Debit",
                  paramFromAcc, paramDebitAmount );
       SqlHelper.ExecuteNonQuery(trans,
            CommandType.StoredProcedure, "Credit",
                  paramToAcc, paramCreditAmount );
       trans.Commit();
       txtResults.Text = "Transfer Completed";
    }
    catch (Exception ex)
    {
       trans.Rollback();
       txtResults.Text = "Transfer Error";
       throw ex;
     }
     finally
    { conn.Close(); } } }
}
```

Per comprendere meglio questo codice occorre conoscere il prototipo del metodo statico ExecuteNonQuery fornito dalla libreria:

```
public static int ExecuteNonQuery(SqlTransaction
           transaction, CommandType commandType,
        string commandText, params SqlParameter[]
                               commandParameters)
```

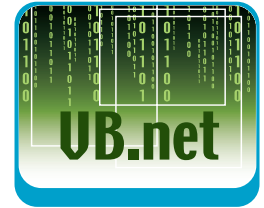
Il metodo accetta una transazione precedentemente aperta ed associata ad una connessione, anch'essa aperta. Gli altri parametri sono gli stessi visti negli esempi precedenti con una particolare menzione per l'ultimo di tipo params. Grazie a questa parola chiave il metodo statico accetta un numero variabile di parametri che verranno automaticamente raggruppati ed inseriti all'interno di un array di tipo SqlParameter.
È per questo motivo che il codice prepara una connessione, la apre e la associa ad un oggetto SqlTransaction. Il metodo utilizza un riferimento ad una transazione già inizializzata e lascia che sia il programmatore a interessarsi della sua gestione.

RICAVARE DATI IN FORMATO XML

L'ultimo metodo fornito dalla libreria permette di ricavare un oggetto XmlReader contenente dati formattati utilizzando il formato XML. Questo grazie anche all'utilizzo della clausola FOR XML fornita da SQL Server che restituisce i record richiesti utilizzando una struttura XML.
Vediamo il codice utilizzato all'interno del pulsante Retrieve XML Data dell'esempio:

```
XmlReader xreader = SqlHelper.ExecuteXmlReader(
  conn, CommandType.Text, "SELECT * FROM Products
                                  FOR XML AUTO");
while (xreader.Read())
{
  txtResults.Text = txtResults.Text +
     xreader.ReadOuterXml() + Environment.NewLine;
}
xreader.Close();
conn.Close();
```

il metodo statico *ExecuteXmlReader* restituisce un oggetto di tipo *XmlReader* riempito con l'XML restituito dalla query specificata come ultimo parametro. Successivamente il metodo Read viene utilizzato per scorrere tra i record ricavati che vengono stampati a video tramite la ReadOuterXml che legge il contenuto del nodo XML corrente, compresi i suoi markup.

Fabio Claudio Ferracchiati

Come sfruttare al massimo l'accesso concorrente ai database

Transazioni Con c# e MySql

Le transazioni sono fondamentali per un uso corretto e sicuro delle basi di dati. In questo articolo capiremo cosa sono e a cosa servono e come sfruttarle in MySQL e in C#

La definizione di transazione nella teoria delle basi di dati è piuttosto noiosa, e recita più o meno così: *"una transazione è una sequenza di istruzioni correlate la cui esecuzione deve essere considerata un'unità indivisibile. Qualora una di tali istruzioni debba essere annullata, devono essere annullate tutte le istruzioni già eseguite nella stessa transazione"*. Nella pratica, è molto più interessante, e ora vediamo il perché. Immaginate di voler prenotare un posto su di un volo e poi in un albergo, per le meritate vacanze dopo aver letto questo articolo. Il computer della vostra agenzia di viaggi dovrà controllare che ci sia posto nel volo e nell'albergo. Per esempio farà una query del tipo:

```
1) select idposto from postiaereo where libero=true
2) select idcamera from albergo where libero=true
3) update albergo...
4) update volo...
```

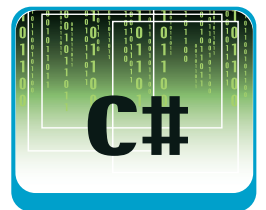
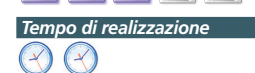
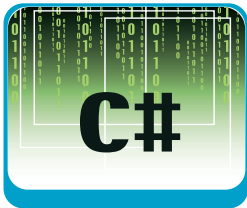
se entrambi i risultati della **1)** e **2)** mostrano posti liberi potrà effettuare la prenotazione tramite le query di update. Poniamo che sull'aereo sia libero un solo posto. Ma che succede se intanto anche un altro lettore chiamiamolo *"Bob"* di questo articolo vuole andare in vacanza e mentre voi siete impegnati nella query **2)** prenota l'ultimo posto sul volo?
A questo punto la vostra agenzia farà l'update, ma non ci saranno posti liberi. Peggio ancora, farà comunque l'update dell'albergo. Risultato netto, avrete prenotato un posto in un albergo che non riuscirete a raggiungere. Immaginate ora che *Bob* non ci sia, ma semplicemente che per qualche motivo l'update alla tabella volo generasse un errore. Il risultato sarebbe sempre lo stesso: una camera d'albergo desolatamente vuota, ma prenotata... Se invece le query **1-2-3-4** fossero eseguibili in modo tale che

- finché non fossero finite nessuno potesse scrivere su quelle tabelle
- se ci fosse una condizione di errore o di anomalia si potesse tornare alla situazione di prima dell'esecuzione

i problemi non si sarebbero verificati. Infatti il posto sull'aereo non sarebbe stato occupato perché l'agenzia do *Bob* non avrebbe potuto scrivere sulla tabella volo. Per eseguire le query in questo modo, abbiamo bisogno di una transazione. In pseudo-codice: *start transaction;* eseguiamo le nostre query.
In caso di errore annulleremo tutti gli effetti delle query fin qui eseguite con il comando: *rollback;* altrimenti renderemo durevoli le modifiche con il comando: *commit.*
Abbiamo visto che le transazioni ci forniscono due aspetti fondamentali:

- Rendono inaccessibili ad altri le tabelle che stiamo modificando. Questo concetto, come vedremo più avanti, è più articolato: per esempio potremmo scegliere di far accedere in sola lettura, ma non in scrittura.
- Ci permettono di annullare gli effetti dei comandi già eseguiti nella transazione qualora lo desiderassimo, per esempio in seguito ad una eccezione.

IN MYSQL... INNODB

Questo articolo si propone di utilizzare le transazioni in MySQL attraverso l'uso di C#. Per prima cosa vediamo quale è il supporto alle transazioni in MySQL. In questo DBMS ogni tabella che creiamo è gestita da un motore di memorizzazione, che ne definisce il tipo. Il tipo più veloce, e quello di default per le versioni di MySQL precedenti alla 4.1 è il tipo *myISAM.* Purtroppo le tabelle *myISAM* non supportano

NOTA

BELLE, MA ACIDE

Il modello di transazioni che viene più utilizzato, e che utilizzeremo con MySQL, è il modello ACID. ACID è l'acronimo di

ATOMICITÀ: le transazioni sono atomiche e indivisibili: o le istruzioni vengono eseguite tutte, o nessuna.
CONSISTENZA: si passa da uno stato valido ad un altro stato valido
ISOLAMENTO: le transazioni non interferiscono fra di loro.
DURABILITÀ: quando una transazione è finita, i suoi effetti sono durevoli

Le transazioni in MySQL rispettano queste quattro proprietà.

le transazioni. Invece il tipo *InnoDB* supporta transazioni ACID. *InnoDB* è il tipo di default su MySQL 4.1, ovvero ogni tabella creata sarà di tipo *InnoDB* se non altrimenti specificato. Se invece sviluppiamo DB per versioni precedenti di MySQL abbiamo due possibilità. La prima è quella di impostare il tipo della tabella al momento della sua creazione, sia tramite i tool visuali ora molto comuni nello sviluppo MySQL, sia attraverso l'SQL:

```
create table impiegato
(IDImpiegato int not null auto_increment primary key,
        nome varchar(255), cognome varchar(255),
  salario int not null bonus int not null) type=InnoDB;
specificando "type=InnoDB"
```

l'altra possibilità è quella di inserire la voce

```
default-storage-engine=INNODB
```

nel file di configurazione del nostro MySQL (generalmente *my.ini*). In questo modo ogni nuova tabella sarà di tipo *InnoDB* se non specificato altrimenti. Le tabelle *InnoDB* supportano quattro tipi di isolamento. Eccoli, ordinati dal più forte al più debole:

- **SERIALIZABLE -** Il massimo per sicurezza e pulizia. Le letture e scritture avvengono in sequenza anche in transazioni che vengono eseguite non in sequenza, visto che tutte le transazioni sono completamente isolate. È però il meccanismo più lento.
- **REPEATABLE READ (default) -** Ogni transazione agisce in una versione completamente isolata della tabella in cui ogni riga non cambia per tutta la transazione. Per questo la lettura è detta ripetibile. Improbabile, ma possibile, avere "righe fantasma": ovvero se si fa una query con una condizione due volte, ma fra le due letture un'altra transazione effettua delle scritture, per la seconda lettura si potrebbero avere delle nuove righe, le righe fantasma.
- **READ COMMITTED -** Le transazioni non sono più isolate, nel senso che le letture non sono necessariamente ripetibili all'interno della stessa transazione.
- **READ UNCOMMITTED -** In questo caso le transazioni… non sono più transazioni, perché non sono più né isolate né consistenti. Abbiamo "letture sporche".

per impostare il livello di isolamento, possiamo specificare uno dei tipi nel comando

```
set transaction isolation level
```

ad esempio

```
set transaction isolation level serializable;
```

oppure modificare la chiave opportuna nel file di configurazione *my.ini:*

```
[mysqld]
transaction-isolation = {READ-UNCOMMITTED
           | READ-COMMITTED | REPEATABLE-READ
                              | SERIALIZABLE}
```

Nella **Tabella 1** riassiumiamo le differenze tra i vari modelli rispetto alle possibili letture. Per ulteriori informazioni potete consultare il manuale online di MySQL all'indirizzo: *http://dev.mysql.com/doc/mysql/en/innodb-transaction-isolation .html*

	Lettura sporca	Lettura non ripetibile	Phantom rows
SERIALIZABLE	No	No	No
REPEATABLE READ	No	No	Sì (improbabile)
READ COMMITTED	No	Sì	Sì
READ UNCOMMITTED	Sì	Sì	Sì

Tabella 1: I vari tipi di tabella InnoDB

INSTALLARE MYSQL, I TOOL E IL MYSQL CONNECTOR.NET

Per poter provare il codice illustrato in questo articolo, dovremo installare MySQL e un provider ADO.NET che ci permetta di accedere ai dati. Potete trovare l'ultima versione di MySQL sul CD allegato alla rivista, oppure scaricarla online all'indirizzo http://dev.MySQL.com/ **Allo stesso indirizzo, nella sezione downloads, scaricare:**
• l'ultima versione del MySQL Query Browser, il versatile strumento di interrogazione di MySQL
• l'ultima versione del MySQL Administrator, la console grafica di amministrazione delle istanze MySQL installate
• l'ultima versione del MySQL Connector/NET, il provider ADO.NET fornito da MySQL.

L'installazione dei pacchetti è semplice, visto che tutti sono forniti di un installer guidato.

TRANSAZIONI IN SQL

Facciamo un esempio di creazione di transazione in SQL. Vogliamo dare un bonus di 1000 euro ad un nostro impiegato (con *id=1*) perché ha imparato ad usare le transazioni, prenderemo i 1000 euro dalla budget di gennaio della cassa aziendale. Dato che entrambe le operazioni debbono andare a buon fine per non rimetterci dei soldi di tasca nostra, potremmo utilizzare una transazione. Faremo riferimento alle tabelle create nella procedura a passi che correda l'articolo. Cominciamo la transazione

```
start transaction;
```

prendiamo i soldi dalla cassa

```
update cassa set budget=budget-1000 where mese=1;
```

tanto che ci siamo, insigniamo il bravo impiegato del titolo di impiegato del mese di gennaio

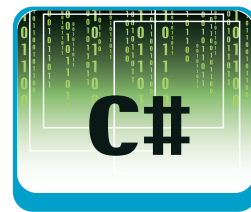
```
insert into  meriti (impiegatodelmese, mese) values(1,1);
```

diamo i soldi al bravo impiegato.

```
update impiegato set bonus= bonus+1000 where
                                   IDImpiegato=1;
```

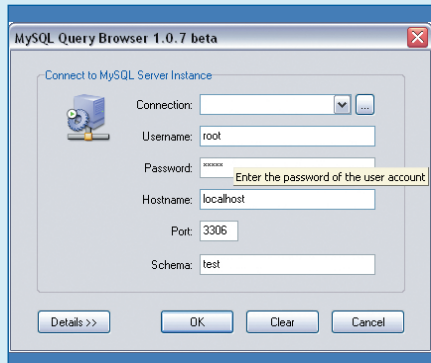
se tutto è andato bene, chiudiamo la transazione

```
commit transaction;
```

se invece la terza update non fosse andata a buon fine, magari perché intanto l'impiegato 1 si è licenziato e ha aperto una sua società basata sulle transazioni, dovremmo annullare le prime due update. Per far questo faremo il rollback della transazione: *rollback;* e tutto tornerà come prima dell'inizio della transazione.

TRANSAZIONI CON C#

I nostri programmi però non sono scritti in SQL, ma in C#. Per poter accedere a MySQL da .NET dovremo utilizzare un provider ADO.NET. Utilizzeremo il connector .NET fornito proprio da MySQL, che potete scaricare dagli indirizzi riportati nel box *Sul Web*. Utilizzeremo Microsoft Visual Studio 2003 per le nostre prove, ma il codice scritto varrà chiaramente per qualunque ambiente .NET. Per prima cosa dobbiamo creare un'applicazione WinForms C#, e aggiungere la *Reference* al *MySQL Connector/NET.* Nelle ultime versioni la dll *MySql .Data.dll* viene installato nella cartella *C:\Programmi\MySQL\MySQL Connector Net 1.0.4\bin\.NET 1.1* o in una equivalente.

CREAZIONE DI TABELLE CHE SUPPORTANO LE TRANSAZIONI

APERTURA DEL QUERY BROWSER

1 **Dopo aver installato tutto il necessario siamo pronti per creare le nostre tabelle *InnoDB.* Per prima cosa dovremo far partire il *MySQL Query Browser* e fare il login.**

CREIAMO IL DATABASE

```
create database ProvaInnoDB;
```

2 **Per creare il nuovo schema di database su cui faremo le nostre prove, possiamo far partire il MySQL Administrator e creare un nuovo schema, oppure crealo direttamente nel *MySQL Query Browser*. Dopo avere inserito il comando nella finestra in alto premiamo poi il pulsante di esecuzione. Se non abbiamo avuto errori, il database sarà stato creato. Per accertarcene basta cliccare con il pulsante destro del mouse sopra la lista degli schemi che si trova a destra nella finestra del *MySQL Query Browser* e selezionare *Refresh Schemata List.***

SETTIAMO L'INNODB

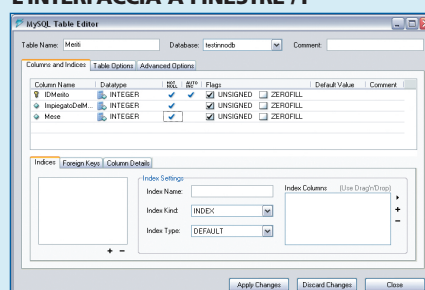
```
default-storage-engine=INNODB
```

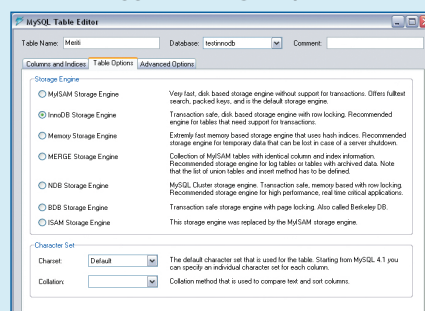
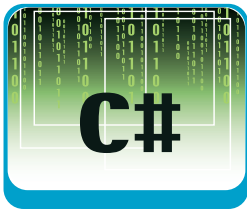
3 **Affinché la tabella che stiamo per creare supporti le transizioni deve essere del tipo *InnoDB.* Se abbiamo installato una versione di MySQL uguale o superiore alla 4.1, questo avviene di default per ogni tabella che andiamo a creare. Per replicare questo comportamento in versioni precedenti (per cui il tipo di default è *MyISAM*) dovremmo andare a modificare come segue una voce del file di configurazione (di solito sotto windows tale file è *my.ini* e si trova nella directory di installazione di MySQL). Vedremo comunque nel prossimo passo come scegliere di volta in volta il tipo della tabella.**

CREIAMO LA TABELLA CON L'SQL

```
use ProvaInnoDB;
create table Impiegato
( IDImpiegato int not null auto_increment
  primary key, Nome varchar(200), Salario
      int not null,Bonus int) type= InnoDB;
insert into impiegato (Nome,Salario,Bonus)
              values ('Luca',1000,100);
insert into impiegato (Nome,Salario,Bonus)
              values ('Claudio', 1000,100);
```

4 **La pri,a riga definisce il database da usare. Vedremo che il nostro nuovo DB sarà selezionato nel pannello dei DB. Le altre righe creano la tabella. L'ultimo parametro, *type= InnoDB;*, specifica il tipo. Questo comando non è necessario se si è certi che il tipo *InnoDB* è il tipo di default. Per vedere la tabella appena creata nella lista dei DB, scegliere *Refresh Schermata List* dal menu contestuale.**

CREIAMO LA TABELLA CON L'INTERFACCIA A FINESTRE /1

5 **Il *MySQL Query Browser* ci permette di creare tabelle in maniera molto semplice attraverso una interfaccia a finestre. Basta fare click con il pulsante destro sopra uno dei database nel pannello di destra e scegliere *Create NewTable*. In figura vediamo la creazione di una tabella attraverso questo strumento. Lo stesso strumento può essere usato per modificare una tabella.**

CREIAMO LA TABELLA CON L'INTERFACCIA A FINESTRE /2

6 **Selezionando la linguetta *TableOptions* è poi possibile specificare che intendiamo creare una tabella *InnoDB.***

Il *Connector* mette a disposizione dello sviluppatore la classe *MySqlTransaction*. Gli oggetti di tipo *MySqlTransaction* rappresentano una transazione, e possono essere associati ai comandi che vogliamo includere nella transazione. Dopo aver letto i paragrafi precedenti, appaiono chiari gli scopi dei membri della classe **Tabella 2**.

A questo punto possiamo procedere a scrivere il codice. Possiamo creare un *Button* sulla form appena creata, e inserire il codice nell'evento *OnClick()*. Per prima cosa specifichiamo che intendiamo utilizzare il namespace adatto, scrivendo in testa al file:

```
using MySql.Data.MySqlClient;
```

Membro	Tipo	Scopo
Connection	Property	*Rappresenta l'oggetto MySqlConnection associato alla transazione*
IsolationLevel	Property	*Rappresenta il livello di isolamento della transazione. Di tipo System.Data.IsolationLevel*
Commit	Method	*Esegue il commit della transazione, rendendo durevoli gli effetti dei comandi*
Rollback	Method	*Esegue il rollback della transazione, annullandone gli effetti.*

Tabella 2: I membri della classe MySQLTransaction

Ora creiamo la connessione (specificando l'opportuna stringa di connessione, in cui inserirete i vostri dati):

```
string ConnString="Database=ProvaInnoDB;Data Source=
              localhost;User Id=root;Password=marco ";
   MySqlConnection Connessione = new
                        MySqlConnection(ConnString);
   Connessione.Open();
```

siamo pronti per creare il *Command* e la transazione. Possiamo poi far partire la transazione e assegnarla al *Command:*

```
   MySqlCommand Cmd = Connessione.CreateCommand();
   MySqlTransaction Transazione;
   Transazione = Connessione.BeginTransaction();
   Cmd.Connection = Connessione;
   Cmd.Transaction = Transazione;
```

mettiamo in un blocco *try* le operazioni sui dati:

```
try
{  int righeModificate=0;
   Cmd.CommandText = "update cassa set budget=
                        budget-1000 where mese=1";
   righeModificate= Cmd.ExecuteNonQuery();
   Cmd.CommandText = "insert into  meriti (
              impiegatodelmese, mese) values(1,1);";
   righeModificate=Cmd.ExecuteNonQuery();
   Cmd.CommandText = "update impiegato set bonus=
                 bonus+1000 where IDImpiegato=1;";
   righeModificate= Cmd.ExecuteNonQuery();
   Transazione.Commit();
}
```

Il *Command* viene eseguito, e ritorna il numero di righe modificate. Se non c'è errore, alla fine facciamo il *Commit()* della transazione. Per rifarci all'esempio SQL sopra illustrato, potremmo sostituire la riga

```
Transazione.Commit();
```

con la seguente, che controlla che l'impiegato da premiare sia ancora fra noi…

```
if (righeModificate==1)
{ Transazione.Commit();}
else
{ Transazione.Rollback();
  MessageBox.Show("Nessuna modifica effettuata");
}
```

È opportuno notare che se si mettesse un breakpoint appena prima del *Commit()*, e si andasse con il Query Browser a fare la *select* sulle tabelle *Impiegato* e *Meriti*, non si rileverebbero cambiamenti, visto che la transazione non ha ancora effettuato le sue modifiche. L'altro caso in cui non vorremmo che la transazione scrivesse niente di durevole nel DB è quando ci fosse un errore. Se per esempio le prime due *ExecuteNonQuery* andassero a buon fine mentre la terza sollevasse un'eccezione, avremmo perso 1000 euro, spariti nel nulla! Occorre allora fare il *RollBack* nel blocco *catch:*

```
catch(Exception exc)
{ try
  { Transazione.Rollback();}
  catch (MySqlException ex)
  { if (Transazione.Connection != null)
    { MessageBox.Show("l' eccezione durante il
                   rollback! " + ex.GetType());} }
  MessageBox.Show("Nessuna modifica effettuata per
              l'eccezione: " + exc.GetType() );}
finally
{Connessione.Close();}
```

Dovremo controllare anche che non ci sia stata nessuna eccezione durante il metodo *RollBack()*.

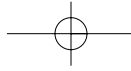
Alla fine chiuderemo comunque la connessione.

CONCLUSIONI

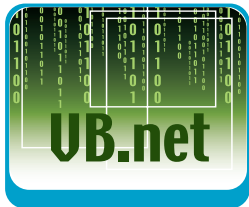
Le transazioni sono indispensabili quando il processo da implementare coinvolge scritture e letture in tempi diversi e su diverse tabelle, ovvero nella maggior parte dei progetti che lo sviluppatore professionista si trova ad affrontare. Buona programmazione!

Marco Poponi

SUL WEB

Si noti che per le versioni di MySQL precedenti alla 4.0 occorre abilitare il supporto *InnoDB* attraverso l'impostazione manuale di opportune chiavi della sezione *[MySQLd]* del file di configurazione *my.ini* o *my.cnf;* è opportuno in tal caso fare riferimento al documento http://dev.MySQL.com/doc/MySQL/en/innodb-in-MySQL-3-23.html **della guida di MySQL.**

La guida definitiva per comprendere a fondo ADO

Accesso ai DB con Visual Basic .Net 2003

L'ultimo articolo che ci aiuterà ad interagire con i database, utilizzando gli strumenti di sviluppo proposti da Visual Basic .Net 2003. Impareremo come salvare, cancellare e modificare i dati

Utilizza questo spazio per le tue annotazioni

La prima azione da eseguire quando si vuole utilizzare una fonte dati è quella di aprire una connessione verso quest'ultima, ciò significa creare un oggetto *SqlConnection*, impostare la stringa di connessione, tramite la proprietà *ConnectionString*, ed aprire la connessione tramite il metodo *Open*. Per operare in modalità connessa, questi punti devono essere eseguiti soltanto alla partenza dell'applicazione, mentre alla fine dell'applicazione si deve chiudere la connessione tramite il metodo *Close*. Per operare in modalità disconnessa, invece, i punti precedenti devono essere eseguiti ogni volta che si deve accedere al database. In pratica, ogni volta che si deve accedere ad un database si devono scrivere le seguenti istruzioni:

```
Dim ObjConnection As New SqlConnection()
ObjConnection.ConnectionString = "Initial
                    Catalog=MiaRubrica;Data
 Source=localhost;Integrated Security=SSPI;"
ObjConnection.Open()
'operazioni su database
ObjConnection.Close()
```

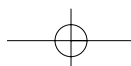
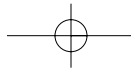
L'OGGETTO SQLCOMMAND

Per interagire con un database, VB.Net mette a disposizione l'oggetto *SqlCommand*. L'oggetto *SqlCommand* deve essere associato ad un oggetto *SqlConnection*, preventivamente connesso alla sorgente dati, e può contenere una query di selezione (per leggere i dati dal database) o una query d'azione (per aggiornare i dati), impostata tramite la proprietà *CommandText*.
Dopo aver impostato l'oggetto *SqlCommand*, ed aver impostato la query mediante la proprietà *CommandText*, si deve utilizzare uno dei relativi metodi *Execute:*

- **ExecuteNonQuery** si utilizza per inviare una query d'azione, e restituisce il numero di record coinvolti. Permette attività di manipolazione di dati, corrispondenti alle istruzioni SQL di Insert, *Update* e *Delete*.
- **ExecuteReader** si utilizza per inviare una query di selezione, e restituisce l'oggetto *DataReader* che consente di accedere al set dei risultati *(resultset)*.
- **ExecuteScalar** si utilizza per inviare una query di selezione, e restituisce un singolo valore risultato di un'istruzione *Select* (da usare, ad esempio, se il risultato è il frutto di query con clausole di aggregazione come *count* o *sum*).

L'oggetto *SqlCommand* espone diverse proprietà, tra queste:

- **CommandText** permette di impostare l'istruzione SQL della query o la stored procedure da eseguire sull'origine dati.
- **CommandType** permette di impostare il valore che indica in che modo interpretare il tipo di query impostato nella proprietà *CommandText*, può assumere i valori: *Text, StoredProcedure*
- **Connection** rappresenta l'oggetto *SqlConnection* associato all'istanza corrente dell'oggetto *SqlCommand*.
- **CommandTimeout** permette di impostare il tempo di attesa (espresso in secondi) trascorso il quale l'esecuzione della query viene annullata e viene sollevata un'eccezione. Per default il suo valore è pari a 30 secondi.
- **Transaction** rappresenta l'oggetto *Transaction* corrispondente alla transazione in cui viene eseguito l'oggetto *SqlCommand*.
- **Parameters** contiene la collezione di tutti i parametri principali associati all'oggetto *SqlCommand*.

Sono inoltre disponibili ulteriori metodi:

- **Cancel** tenta di annullare l'esecuzione dell'oggetto *SqlCommand.*
- **CreateParameter** crea un oggetto *Parameter* associato al comando parametrico in questione.
- **ResetCommandTimeout** reimposta la proprietà *CommandTimeout* al valore predefinito (30 secondi).

INSERIRE O MODIFICARE UNA PERSONA

Per eseguire una query di azione sul database, tipicamente si deve:

- Creare un oggetto *SqlCommand* con le proprietà fondamentali *CommandText* e *SqlConnection.*
- Eseguire il comando *Sql* tramite il metodo *ExecuteNonQuery,* che restituisce il numero di record coinvolti dall'istruzione.

Inseriamo, ad esempio, una nuova persona nel database *MiaRubrica.*
La prima operazione è quella di costruire la stringa corrispondente all'istruzione Sql di inserimento *(Insert)*

```
Dim QuerySql As String
QuerySql = "INSERT INTO Persona (CodicePersona,
                Nome,Cognome,NumeroTelefono)VALUES
( 1,'Federica','Buono','09888888')"
```

Successivamente si deve creare un oggetto *SqlCommand* ponendo la proprietà *CommandText* pari alla stringa che definisce la query, e la proprietà *SqlConnection* pari all'oggetto *objConnection* creato nell'esempio precedente:

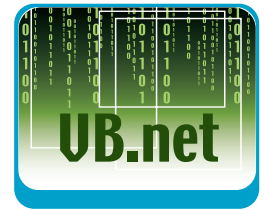
```
Dim ObjCommand As New SqlCommand(QuerySql,
                                 ObjConnection)
```

Infine si deve eseguire il comando Sql tramite il metodo *ExecuteNonQuery,* che restituisce il numero di record coinvolti dall'istruzione:

```
Dim NumeroRecord As Integer
NumeroRecord = ObjCommand.ExecuteNonQuery()
```

Seguendo lo stesso schema possiamo modificare un record della tabella persona, costruendo la stringa corrispondente all'istruzione Sql di aggiornamento *(update)*

```
Dim QuerySql As String
QuerySql = "update Persona set Nome ='Sara',
        Cognome='Buono', NumeroTelefono='5556666'
where CodicePersona=1"
Dim ObjCommand As New SqlCommand(QuerySql,
                                 ObjConnection)
Dim NumeroRecord As Integer
NumeroRecord = ObjCommand.ExecuteNonQuery()
```

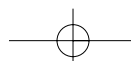
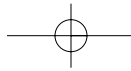
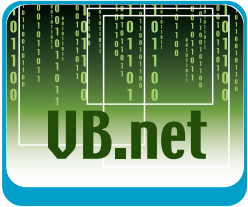
Per eliminare una persona, si deve seguire sempre lo stesso schema utilizzando l'istruzione Sql di cancellazione *(delete Persona where CodicePersona=1).*
Quando si effettuano operazioni di *INSERT, UPDATE* e *DELETE* sulle tabelle del database, diventa conveniente utilizzare l'oggetto *SqlCommand* con le Stored Procedure

LE STORED PROCEDURE

Le stored procedure sono degli oggetti di database, simili a dei file batch di istruzioni T-Sql, che possono essere riutilizzati e richiamati avvalendosi semplicemente del nome della stored procedure stessa. Sono spesso utilizzate come contenitore per la logica di business di un applicazione che utilizza database.
La manipolazione di dati è senz'altro il punto in cui le stored procedure mostrano, la più ampia utilità. Tra i vantaggi del loro utilizzo, è da segnalare la velocità di esecuzione delle varie istruzioni, infatti la prima volta che queste vengono utilizzate, si forma un "piano di esecuzione" della stored procedure che al successivo accesso risulterà molto più rapido nell'esecuzione dello script. Tutto questo è molto più rapido rispetto all'esecuzione delle singole istruzioni che avrebbero un piano d'accesso ripetuto per ogni istruzione. Le stored procedure comunicano, con l'applicazione che esegue le chiamata, tramite i propri parametri. I parametri di una stored procedure sono analoghi agli argomenti di una routine VB e

NOTA

Per interrogare il database e leggere i dati si può utilizzare anche il metodo ExecuteScalar. Il metodo ExecuteScalar consente di eseguire in modo efficiente una query che restituisce un unico valore scalare, per questo può essere utilizzato per leggere il risultato di fun-zioni di aggregazione quali ad esempio Max o Sum.

MODALITÀ CONNESSA E MODALITÀ DISCONNESSA

La domanda: "è meglio operare in modalità connessa oppure in modalità disconnessa?" è da sempre oggetto di discussioni filosofiche tra sviluppatori, cerchiamo di capire il perché. Nelle tradizionali applicazioni client/server, si stabilisce una connessione ad un database all'avvio dell'applicazione e si mantiene attiva durante tutto il ciclo di vita dell'applicazione. Questo approccio diminuisce il tempo di attesa delle operazioni sul database poiché elimina il tempo necessario alla connessione, ma in alcuni casi non è possibile da utilizzare (Risorse di sistema limitate, Gestione limitata di connessioni simultanee del database, Applicazioni web). In modalità disconnessa, in pratica, si apre una connessione, solo quando serve, si estrae un blocco di dati, si memorizza sul client e quindi si chiude la connessione per rilasciare le risorse legato server ad essa associate. Dopo aver caricato i dati sul client, è possibile utilizzarli per eseguire qualsiasi tipo di elaborazione, come per esempio aggiungere nuovi record oppure modificare o eliminare quelli esistenti. Una volta che sono terminate le opportune elaborazioni sui dati, si riapre la connessione, si scaricano i dati sul server e si richiude di nuovo. Le applicazioni sono quindi connesse al database solo il tempo necessario per recuperare o aggiornare i dati. Di contro aprire e chiudere la connessione significa maggiore tempo di attesa.

possono essere utilizzati come variabili standard nel linguaggio di programmazione Transact-SQL. I parametri devono iniziare con il carattere @ e possono essere di input e di output, attraverso i parametri di output è possibile stabilire come è terminata l'esecuzione di una stored procedure. La creazione di una stored procedure avviene tramite l'istruzione CREATE PROCEDURE e consta di una parte dichiarativa, ed una parte in cui viene specificato il codice T-Sql. La parte dichiarativa iniziale, contiene il nome della stored procedure e la dichiarazione dei parametri, facoltativi, che si occupano di interfacciare la stored procedure con l'applicazione esterna. La seconda parte della stored procedure, cioè quella che contiene il codice T-Sql, viene introdotta dalla parola chiave AS che si occupa appunto della separazione tra la dichiarazione dei parametri ed il successivo codice T-Sql. Ad esempio, la stored procedure che esegue l'inserimento di un record nella tabella Persona, si può scrivere:

```
CREATE PROCEDURE [insert_Persona_1]
        (@CodicePersona_1        [int],
         @Nome_2          [varchar](50),
         @Cognome_3       [varchar](50),
         @NumeroTelefono_4        [varchar](20))
AS INSERT INTO [MiaRubrica].[dbo].[Persona] (
  [CodicePersona], [Nome], [Cognome],[NumeroTelefono])
 VALUES
   ( @CodicePersona_1, @Nome_2,
                @Cognome_3, @NumeroTelefono_4)
```

UTILIZZARE LE STORED PROCEDURE

Siamo pronti per utilizzare una stored procedure in VB.Net. Tipicamente si deve utilizzare l'oggetto SqlCommand impostando:

- La proprietà **CommandText** al nome della Stored Procedure
- La proprietà **CommandType** al valore StoredProcedure

Ad esempio, per eseguire la Stored Procedure che si occupa di inserire una nuova persona nel database, possiamo scrivere:

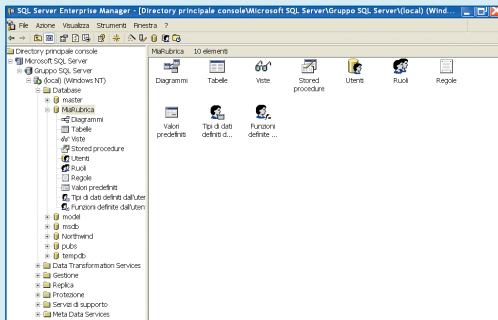
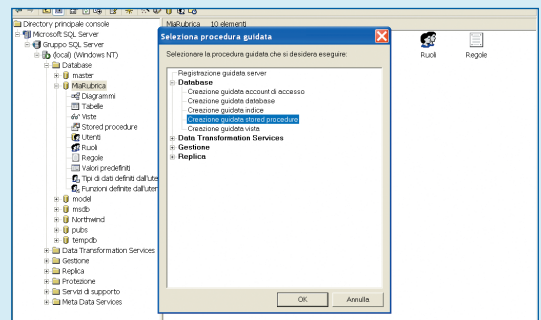
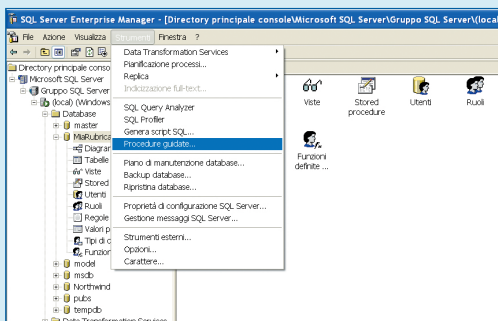
```
Dim ObjCommand As SqlCommand
```

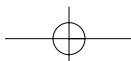
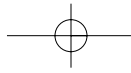
NOTA

Per mantenere il codice il più compatto possibile, si può utilizzare l'istruzione Imports che semplifica l'accesso alle classi, eliminando la necessità di digitare in modo esplicito il nome completo del namespace che le contiene. Le istruzioni Imports devono sempre essere scritte nella parte superiore del file nel quale si vogliono utilizzare, prima di qualunque altro codice. Per queste ragioni, in tutti gli esempi di codice, è ragionevole assumere l'inserimento delle seguenti istruzioni Imports:

```
Imports System.Data
Imports System.Data
.SqlClient
```

CREAZIONE GUIDATA STORED PROCEDURE

Ecco i passi necessari per creare le Stored Procedure di inserimento, modifica e cancellazione su una tabella in SQL Server 2000

1 La prima operazione da compiere è quella di espandere il ramo del Server fino alla cartella Database e selezionare il Database d'esempio (*MiaRubrica*)

3 Dalla finestra Seleziona Procedura guidata, si deve espandere la voce: *Database* e selezionare l'opzione: *Creazione guidata stored procedure*

2 Selezionare il menu: StrumentiÆProcedure Guidate

4 Dalla schermata di ingresso si deve cliccare su Avanti

```
ObjCommand = New SqlCommand("insert_Persona_1",
                                    ObjConnection)
ObjCommand.CommandType =
                       CommandType.StoredProcedure
```

Naturalmente le istruzioni appena scritte non sono sufficienti, poiché dobbiamo valorizzare i campi della tabella persona, per questo scopo possiamo utilizzare i parametri. L'oggetto SqlCommand può contenere una collezione di oggetti SqlParameter che rappresenta ognuno un diverso parametro della stored procedure. Il nome del parametro deve corrispondere al nome definito nella Stored Procedure e viene passato come primo argomento al costruttore dell'oggetto Parameter. In generale, il tipo di ogni oggetto Parameter deve corrispondere a quello del parametro accettato dalla stored procedure. È possibile passare il tipo come secondo argomento al costruttore dell'oggetto Parameter utiliz-zando un valore enumerato SqlDbType che specifica i tipi di dati SQL Server. Come terzo argomento del costruttore è possibile passare la dimensione del parametro. Infine tramite la proprietà Value si può assegnare un valore al parametro. Continuando l'esempio precedente, per valorizzare i parametri della Stored Procedure di inserimento, si può scrivere:

```
Dim spPar As SqlParameter
spPar = ObjCommand.Parameters.Add(
          "@CodicePersona_1", SqlDbType.Int, 4)
spPar.Value() = 1
spPar = ObjCommand.Parameters.Add("@Nome_2",
                          SqlDbType.VarChar, 50)
spPar.Value() = "Federica"
spPar = ObjCommand.Parameters.Add(
          "@Cognome_3", SqlDbType.VarChar, 50)
spPar.Value() = "Buono"
spPar = ObjCommand.Parameters.Add(
     "@NumeroTelefono_4", SqlDbType.VarChar, 20)
spPar.Value() = "05588888888"
```

Infine, come negli esempi precedenti, si esegue l'istruzione sul database:

```
Dim NumeroRecord As Integer
NumeroRecord = ObjCommand.ExecuteNonQuery()
```

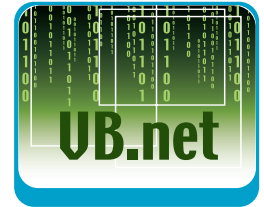
Quando si utilizzano le stored procedure occorre tener conto della direzione di ciascun parametro. Se si invoca una stored procedure, che restituisce un valore attraverso un argomento, è necessario impostare la proprietà Direction ad InputOutput oppure ad Output. Per default tutti i parametri vengono creati come parametri di input perciò, in questo caso, non è necessario impostare la proprietà Direction.

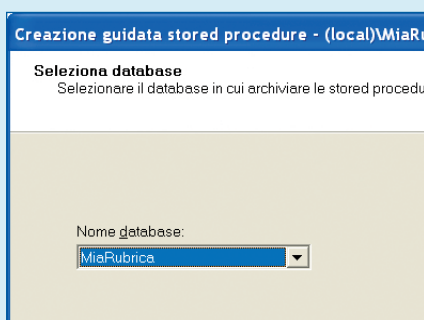
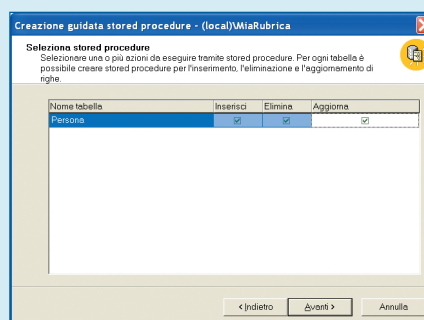
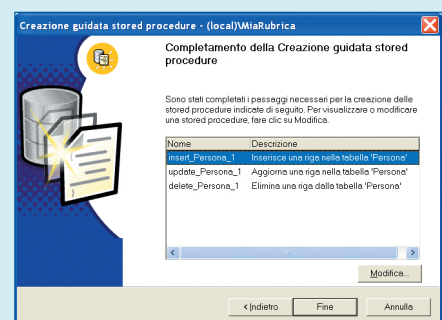
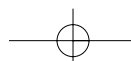
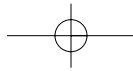
INTERROGARE IL DATABASE

Per interrogare il database e leggere i dati si deve:

- Creare un oggetto SqlCommand come abbiamo visto in precedenza.
- Eseguire la query Sql tramite il metodo ExecuteReader, ed assegnarlo ad un ogget-to SqlDataReader per leggere il set dei risultati una riga per volta.

Ad esempio, per eseguire una ricerca di tutte le persone presenti in MiaRubrica, si può scrivere:

```
Dim QuerySql As String
QuerySql = "select * from persona"
Dim ObjCommand As New SqlCommand(QuerySql,
                                  ObjConnection)
Dim ObjReader As SqlDataReader =
                       ObjCommand.ExecuteReader()
```

NOTA

La proprietà ConnectionString permette di impostare la stringa di connessione al database. La stringa di connessione consente di definire il nome del database di origine ed altri parametri necessari a stabilire la connessione iniziale, delimitati da punto e virgola. Ad esempio l'attributo Data Source può assumere il valore localhost nel caso dobbiamo connetterci ad SQL Server sulla macchina locale, ma in generale deve contenere il nome del computer sul quale è installato il database.

5 **In questa maschera è necessario selezionare il database su cui si devono generare le Stored Procedure dal menu a tendina. Il database *MiaRubrica* dovrebbe essere selezionato di default. Per confermare la scelta è sufficiente cliccare su *Avanti*.**

6 **Selezionare le Stored Procedure che si vogliono creare su ogni tabella del database. Nel nostro caso sarà presente soltanto la tabella *Persona* su cui dovremo selezionare i tre tipi di Stored Procedure da creare (Inserisci, elimina, aggiorna)**

7 **Dall'ultima finestra si possono modificare il nome e la struttura delle stored procedure. È sufficiente selezionare la stored procedure e cliccare sul tasto modifica. Infine cliccando sul tasto *Fine* le tre stored procedure verranno create nel Database**

Analizziamo in dettaglio come utilizzare l'oggetto SqlDataReader per leggere il set di risultati ottenuto.

L'OGGETTO SQLDATAREADER

L'oggetto SqlDataReader viene utilizzato per leggere un set dei risultati di tipo forward-only da un database SQL Server, a seguito di una query d'interrogazione. A differenza degli oggetti visti in precedenza non è possibile utilizzare un costruttore per creare un oggetto SqlDataReader, ma è invece necessario chiamare il metodo ExecuteReader dell'oggetto SqlCommand Tra le proprietà ed i metodi esposti dall'oggetto SqlDataReader segnaliamo:

- **FieldCount** contiene il numero di colonne del record corrente.
- **IsClosed** contiene un valore, pari a *True*, se il *DataReader* è chiuso.
- **RecordsAffected** contiene il numero di record modificati, inseriti o eliminati dall'esecuzione dell'istruzione SQL.
- **Close** chiude l'oggetto *SqlDataReader* e rilascia le risorse allocate.
- **GetName** contiene il nome della colonna con l'indice specificato.
- **IsDbNull** contiene un valore, pari a *True*, se la colonna contiene valori nulli.
- **Read** sposta l'oggetto *SqlDataReader* al record successivo. Restituisce un valore pari a *True* se sono presenti altri record da elaborare, oppure False se si è arrivati alla fine del set dei risultati
- **GetValue** contiene il valore della colonna con l'indice specificato nel suo formato nativo

Script SQL della tabella persona in cui CodicePersona è la primary key

```
CREATE TABLE
[dbo].[Persona] (
[CodicePersona] [int]
NOT NULL ,
[Nome] [varchar] (50)
NULL ,
[Cognome] [varchar]
(50) NULL ,
[NumeroTelefono]
[varchar] (20) NULL
) ON [PRIMARY]
```

Al posto del metodo GetValue è possibile utilizzare uno dei numerosi metodi Get fortemente tipizzati, come GetString o GetDecimal. In pratica c'è un metodo Get per ogni tipo di dati gestito da VB oppure un metodo GetSql per ogni tipo di dati gestito da Sql Server. Questi metodi particolari dovrebbero essere sempre adottati, in quanto evitano gli errori di conversione provocati dalla perdita di precisione e generano codice più veloce.

UTILIZZARE L'OGGETTO SQLDATAREADER

Per iterare sui singoli record di un set di risultati, utilizzando l'oggetto SqlDataReader, è sufficiente:

- invocare il metodo Read per avanzare al record successivo nel set di risultati
- verificare il relativo valore di ritorno per controllare se esistono altri risultati (nel caso sia pari a True) oppure si è arrivati alla fine e non c'è nessun'altra riga da leggere (se è False).

Grazie al funzionamento del metodo Read è possibile creare un ciclo While..End While basato sull'oggetto SqlDataReader. Per completare l'esempio precedente, in cui si cercano tutti le persone presenti in MiaRubrica, possiamo scrivere il seguente codice che mostra un messaggio per ogni persona in archivio:

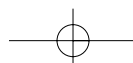
```
Dim CodicePersona, Nome, Cognome, NumeroTelefono
                                          As String
While ObjReader.Read()
    CodicePersona = ObjReader.GetInt32(0)
    Nome = ObjReader.GetString(1)
    Cognome = ObjReader.GetString(2)
    NumeroTelefono = ObjReader.GetString(3)
    MessageBox.Show(CodicePersona & " " & Nome &
            " " & Cognome & " " & NumeroTelefono)
End While
```

Mentre si utilizza l'oggetto *SqlDataReader*, non è possibile eseguire alcuna operazione sull'oggetto *SqlConnection* associato, per questo è importante chiudere l'oggetto SqlDataReader quando non esistono più righe da elaborare, in modo da rilasciare le risorse (lato client e lato server) e rendere la connessione nuovamente disponi-bile per altri comandi:

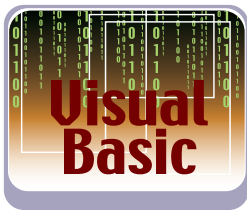
```
ObjReader.Close()
```

CONCLUSIONI

Esistono altri metodi per l'accesso ai dati. Uno di questi é rappresentato dagli oggetti *SqlDataAdapter* e *Dataset* tramite i quali si realizza una sorta di approccio disconnesso ai dat, tale che la view dei dati viene copiata in locale in una forma statica; vengono effettuate tutte le modifiche nexessarie, e una volta compiute le varie operazioni il database viene risincronizzato con quello originale. Delle due modalità fornite da ADO.NET per la gestione dei dati, i resultset forward-only di sola lettura sono quelli che consentono di ottenere le prestazioni migliori. Ciò è dovuto al fatto che il DataSet non mantiene attiva una connessione alla sua origine dati per tutto il tempo in cui l'oggetto esiste, come al contrario accade per l'oggetto SqlDataReader. Mentre SqlDataReader non è scalabile come la combinazione SqlDataAdapter/DataSet, può fornire migliori prestazioni per un'origine dati remota dal momento che essa restituisce dati come un cursore forward-only in sola lettura.
A voi stabilire quale modalità utilizzare a seconda dei casi.

Luigi Buono

Utilizzare Shell32.dll in VB

Undocumented Shell32 library

Vediamo come sfruttare alcune funzionalità non documentate in VB Mostreremo le dichiarazioni e gli esempi d'uso delle funzioni che ci consentono di dialogare con il cuore del sistema operativo

Per shell s'intende quella componente del sistema operativo attraverso la quale è possibile compiere la maggior parte delle operazioni che comuni che "investono" normalmente la vita di tutti: apertura di file, navigazione tra le directory, configurazione di stampanti, creazione di collegamenti sul desktop, ecc. La shell, organizza tutto quest'insieme d'entità all'interno di una struttura gerarchica, meglio conosciuta con il nome di Namespace. La Shell Namespace possiamo immaginarla come equivalente al noto filesystem, malgrado consenta anche la gestione di oggetti speciali come i virtual folder. In questo contesto, è necessario adottare una strategia d'identificazione diversa da quella utilizzata dal "semplice" filesystem ed è per questo che si parla di item ID.

LA FUNZIONE SHRUNDIALOG()

La prima funzione che analizzeremo è la *SHRunDialog()*, una "undocumented function" la cui sintassi in Visual Basic è la seguente:

```
Public Declare Function SHRunDialog Lib "shell32"
        Alias "#61" (ByVal hWnd As Long, ByVal hIcon
        As Long, ByVal sPath As String, ByVal sTitle
        As String, ByVal sPrompt As String, ByVal uFlags
        As Long) As Long
```

Scopo della funzione è quello di mostrare a video una finestra di dialogo attraverso la quale consentire l'avvio di un determinato programma. Per personalizzare il funzionamento e la modalità di richiesta all'utente, viene sfruttato in particolare l'ultimo parametro, *uFlags* che opportunamente settato, permette di nascondere il pulsante Sfoglia oppure la lista dei programmi avviati *(MRU- Most Recently Used)*. Di seguito un piccolo esempio che implementa queste funzionalità:

```
Private Sub cmdRunDialog_Click()
  Dim uFlag As Long
  Dim sTitle, sPrompt, sLw, SysDir As String
  If Option1(0).Value=True Then
    SHRunDialog hWnd,0,0,vbNullString,vbNullString,0
  Else
    If Check1(0).Value=1 Then uFlag=&H3 Else
                                   uFlag=&H2
    If Check1(1).Value=1 Then
      SysDir=Space$(256)
      Ret=GetSystemDirectory(SysDir,Len(SysDir))
      If Ret<> 0 Then _
        SysDir=Left$(SysDir,
               Len(Trim$(SysDir))-1)&"\Shell32.dll"
        hIcon=ExtractIcon(App.hInstance,SysDir,46)
      Else
        hIcon=0
      End If
      sTitle=txtSHRunDialog.Text
      sPrompt=txtPrompt.Text
```

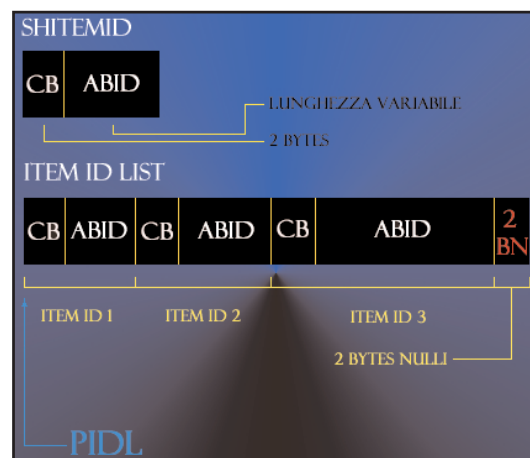



Fig. 1: *Item ID, item ID List e PIDL*

```
        sLw=txtCercaIn.Text
        'Se il sistema operativo è Windows NT like,
        'allora trasforma tutte le stringhe
        'utili in formato Unicode
          If IsWinNT Then
             sTitle=StrConv(sTitle,vbUnicode)
             sPrompt=StrConv(sPrompt,vbUnicode)
             sLw=StrConv(sLw,vbUnicode)
          End If
        'Avvia la finestra di dialogo
          SHRunDialog
            hWnd,hIcon,sLw,sTitle,sPrompt,uFlag
     End If
End Sub
```

Un dettaglio importante e che ritroveremo spesso all'interno delle restanti procedure, è la preventiva conversione in Unicode delle stringhe passate alle API della Shell32 qualora il sistema operativo riscontrato sia Windows NT-like.

GLI SPECIAL FOLDERS

Abbiamo visto che la Shell consente di gestire diversi tipi di oggetti, tra i quali anche quelli cosiddetti virtuali come la cartella dei documenti e quella delle stampati. Questo genere di oggetti fa parte di quell'insieme definito come cartelle speciali tra le quali, oltre a quelli definiti virtuali, ve ne sono altri definiti "standard" come System o Program Files. Ovviamente la lista è molto più lunga e possiamo ricavare il reale percorso fisico sul disco rigido attraverso l'API SHGetSpecialFolderLocation() e SHGetPathFromIDList() le cui sintassi VB sono rispettivamente:

```
Public Declare Function SHGetSpecialFolderLocation Lib
    "shell32" (ByVal hwndOwner As Long, ByVal nFolder
                    As Long, PIDL As Long) As Long
Public Declare Function SHGetPathFromIDList Lib
       "Shell32.dll" Alias "SHGetPathFromIDListA" (ByVal
         PIDL As Long, ByVal pszPath As String) As Long
```

La prima consente di ottenere un puntatore ad un oggetto di tipo ITEMIDLIST, mentre la seconda servendosi di questa informazione, ne ricava il percorso fisico su disco rigido riportandolo all'interno del secondo parametro pszPath. Tralasciando i dettagli e volendo rendere il tutto più "semplice", è sufficiente dire che i passi da compiere sono solo questi:

- passare alla *SHGetSpecialFolderLocation()* la costante *CSIDL* (un identificativo che rappresenta una particolare cartella speciale). Se tutto va a buon fine, essa ci restituirà in PIDL il valore da utilizzare per la seconda chiamata;
- passare a *SHGetPathFromIDList()* il valore di PIDL e, se tutto va bene, ci ritroveremo in *pszPath* il percorso che volevamo.

Di seguito un esempio su entrambe le funzioni:

```
Private Sub cmbSpecialFolder_Click()
      'Visualizza il percorso completo di una cartella
   'speciale sul proprio HD
     Dim Index As Long
Index=cmbSpecialFolder.ItemData(cmbSpecialFolder
                                        .ListIndex)
     If Index>-1 Then
         txtSpecialFolderPath.Text=GetSpecialFolder(Index)
     End If
End Sub
```

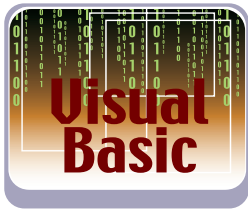
dove:

```
Function GetSpecialFolder(CSIDL As Long) As String
   Dim sPath As String
   Dim PIDL As Long
   If SHGetSpecialFolderLocation(
          frmPrincipale.hWnd,CSIDL,PIDL)=S_OK Then
      'Se il PIDL è correttamente riportato,
      'preleva il percorso dall'ID list
        sPath=Space$(MAX_PATH)
        If SHGetPathFromIDList(
                        ByVal PIDL,ByVal sPath) Then
```

LE COSTANTI CSIDL

Di seguito ecco alcune costanti CSIDL:

Costante	Descrizione
CSIDL_ADMINTOOLS	(user name>\Start Menu\Programs\ Administrative Tools)
CSIDL_APPDATA	(Application Data, new for NT4)
CSIDL_COMMON_ADMINTOOLS	(All Users\Start Menu\Programs\ Administrative Tools)
CSIDL_COMMON_APPDATA	(All Users\Application Data)
CSIDL_COMMON_DOCUMENTS	(All Users\Documents)
CSIDL_COMMON_PROGRAMS	(Programs folder (under Start menu in All Users profile))
CSIDL_DESKTOP	(Desktop (namespace root))
CSIDL_DESKTOPDIRECTORY	(Desktop folder)
CSIDL_FAVORITES	(Favorites folder)
CSIDL_FLAG_CREATE	(new for Win2K, or this in to force creation of folder)
CSIDL_FONTS	(Fonts virtual folder)
CSIDL_HISTORY	(History folder)
CSIDL_INTERNET_CACHE	(Internet Cache folder)
CSIDL_MYPICTURES	(My Pictures, new for Win2K)
CSIDL_NETHOOD	(Network Neighborhood directory)
CSIDL_PERSONAL	(Personal folder)
CSIDL_PROGRAM_FILES	(C:\Program Files)
CSIDL_PROGRAM_FILES_COMMON	(C:\Program Files\Common)
CSIDL_PROGRAMS	(Programs folder (under Start menu); My Documents)
CSIDL_RECENT	(Recent folder)
CSIDL_SENDTO	(SendTo folder)
CSIDL_STARTMENU	(Start menu)
CSIDL_STARTUP	(Startup folder)
CSIDL_SYSTEM	(GetSystemDirectory())
CSIDL_TEMPLATES	(Templates folder)
CSIDL_WINDOWS	(GetWindowsDirectory())

Maggior informazioni possono essere recuperate sul sito della Microsoft al link
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_winshell.asp

```
            GetSpecialFolder=Left(
                        sPath, InStr(sPath,Chr$(0))-1)
        End If
      'Libera il PIDL
        Call CoTaskMemFree(PIDL)
    End If
End Function
```

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

LE FUNZIONI SHRESTARTSYSTEMMB() E SHSHUTDOWNDIALOG()

Le due funzioni che stiamo per mostrare sono molto simili tra loro e per questa ragione le tratteremo all'interno dello stesso paragrafo. La *SHRestartSystemMB()* è una funzione non documentata ed il suo scopo è semplicemente quello di effettuare il riavvio, il logoff, ecc… del sistema.
La sua sintassi è la seguente:

```
Public Declare Function SHRestartSystemMB Lib
    "shell32" Alias "#59" (ByVal hOwner As Long, ByVal
      sPrompt As String, ByVal uFlags As Long) As Long
```

dove:

- **hOwner:** handle dell'owner window (tipicamente la form principale);
- **sPrompt:** testo aggiuntivo che apparirà a video per la conferma dell'operazione;
- **uFlags:** costante che identifica il tipo di operazione da effettuare.

Vediamo subito il codice relativo a questa funzionalità:

```
Private Sub cmdRestartDialog_Click()
   Dim ExtraMsg As String
   Dim uFlag As Long
   ExtraMsg = TxtRestartPrompt
   If IsWinNT Then ExtraMsg = StrConv(ExtraMsg,
vbUnicode)
   Select Case cmbRestartOp.ListIndex
          Case 0:  uFlag = EWX_LOGOFF
          Case 1:  uFlag = EWX_SHUTDOWN
          Case 2:  uFlag = EWX_REBOOT
          Case 3:  uFlag = EWX_FORCE
          Case 4:  uFlag = EWX_POWEROFF
   End Select
   If SHRestartSystemMB(frmPrincipale.hWnd,
                              ExtraMsg, uFlag) = vbYes Then
      MsgBox "OK... alla prossima volta..."
   End If
End Sub
```

Anche la *SHShutDownDialog()* non è documentata e la sua sintassi risulta essere la seguente:

```
Public Declare Function SHShutDownDialog Lib
      "shell32" Alias "#60" (ByVal Flag As Long) As Long
```

Accetta un solo parametro identificato da una delle costanti precedentemente utilizzate dalla *SHRestartSystemMB()*. Il suo unico scopo è quello di mostrare la finestra di chiusura del sistema allo stesso modo di quanto avviene quando l'utente decidere di spegnere il proprio PC.

LA STRUTTURA SHFILEOPSTRUCT

Di seguito sono mostrati i principali valori che può assumere l'item fFlags della struttura *SHFILEOPSTRUCT*

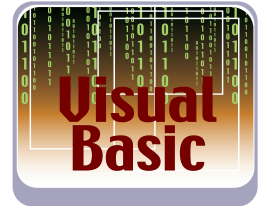
FOF_CREATEPROGRESSDLG	La finestra di dialogo è visibile.
FOF_MULTIDESTFILES	Indica che pTo specifica destinazioni multiple per i file.
FOF_CONFIRMMOUSE	Non implementato
FOF_SILENT	La finestra di dialogo non è visibile.
FOF_RENAMEONCOLLISION	Se nella directory di destinazione, esiste un file con un nome identico al sorgente, viene creato un nuovo file del tipo "Copia1 di ..."
FOF_NOCONFIRMATION	Non chiede conferma all'utente
FOF_WANTMAPPINGHANDLE	La funzione SHFileOperation riempe hNameMappings (leggere più avanti)
FOF_ALLOWUNDO	Preserva le informazioni per un eventuale UNDO.
FOF_FILESONLY	Nel caso si specifichi *.* effettua l'operazione solo sui file e ne migliora le prestazioni.
FOF_SIMPLEPROGRESS	La finestra di dialogo è visibile, ma non vengono visualizzati i nome dei file.
FOF_NOCONFIRMMKDIR	Nel caso sia necessario creare una nuova directory, il sistema non chiede conferma.
FOF_NOERRORUI	Nessun messaggio in caso di errori
FOF_NORECURSION	Nessuna ricorsione nelle sottodirectory. L'operazione avviene solo a livello di directory corrente
FOF_NOCOPYSECURITYATTRIBS	Non copia gli attributi di sicurezza
FOF_NO_CONNECTED_ELEMENTS	(vedi riquadro I Connecting File)
FOF_WANTNUKEWARNING	Avverte l'utente con un warning nel caso si stia eliminando un file (non nel Cestino).
FOF_NORECURSEREPARSE	Nessuna ricorsione sui reparse point

LA FUNZIONE SHBROWSEFORFOLDER()

La funzione SHBrowseForFolder() mostra a video una finestra di dialogo che ci consente di scorrere le cartelle ed i file del nostro disco rigido, con la possibilità d'impostare dei filtri sugli oggetti visualizzati e selezionabili. La sintassi è:

```
Public Declare Function SHBrowseForFolder Lib
                 "shell32" (lpbi As BROWSEINFO) As Long
```

La struttura *BROWSEINFO* è così dichiarata:

```
Public Type BROWSEINFO
      hwndOwner As Long
```

```
        pidlRoot As Long
        pszDisplayName As String
        lpszTitle As String
        ulFlags As Long
        lpfn As Long
        lParam As Long
        iImage As Long
End Type
```

In particolare, il parametro ulFlags definisce proprio il filtro per l'esplorazione delle cartelle. Il valore di questo item della struttura può essere una combinazione di svariate costanti (identificate con prefisso BIF). Vediamo un esempio:

```
Private Sub cmdBrowseFolder_Click()
   Dim iNull As Integer
   Dim IDList As Long, Result As Long
   Dim sPath As String
   Dim BrwInf As BROWSEINFO
   'Inizializza la struttura
     With BrwInf
          .hwndOwner=Me.hWnd
          .pszDisplayName=Space(260)
          .lpszTitle=("Shell32 API - by ioProgrammo")
          .ulFlags=BIFoption
     End With
   IDList=SHBrowseForFolder(BrwInf)
   If IDList Then
      sPath=String$(MAX_PATH,0)
      SHGetPathFromIDList IDList,sPath
      'Libera IDList
        CoTaskMemFree IDList
      iNull=InStr(sPath,vbNullChar)
      If iNull Then
         sPath=Left$(sPath, iNull - 1)
      End If
   End If
   'Ritorna le informazioni
     txtPath.Text=sPath
     txtSelectedFolder.Text=BrwInf.pszDisplayName
     txtIdxImage.Text=(BrwInf.iImage)
End Sub
```

LA FUNZIONE SHCHANGEICONDIALOG()

La funzione in oggetto ha la seguente sintassi:

```
Public Declare Function SHChangeIconDialog Lib
    "shell32" Alias "#62" (ByVal hOwner As Long, ByVal
          szFilename As String, ByVal Reserved As Long,
                          lpIconIndex As Long) As Long
```

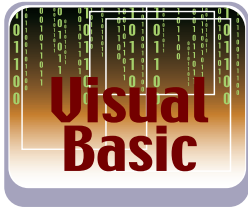
La funzione non fa altro che mostrare a video una finestra di dialogo attraverso la quale possiamo scegliere un'icona dal file considerato ritornando l'indice della stessa all'interno del parametro lpIconIdx.

```
Private Sub cmdChangeIcon_Click()
  Dim FileName As String
  Dim IconIdx As Long
  Dim SmallIcon As Long, LargeIcon As Long
  'Alloca il buffer
    FileName=txtIconPath&String$(MAX_PATH - Len(
                                     txtIconPath),0)
  'Se il sistema operativo è Windows NT-like, allora
  'converti la stringa in formato Unicode
    If IsWinNT Then FileName=StrConv(
                                 FileName,vbUnicode)
  IconIdx=Val(txtIconIdx)
  'Il valore di ritorno della funzione è:
  '0: l'utente ha cancellato l'operazione
  '1: selezione effettuata
    If SHChangeIconDialog(
                      hWnd,FileName,0,IconIdx) Then
      'Mostra la selezione fatta
        txtIconPath=GetStrFromBuffer(FileName)
        txtIconIdx=IconIdx
       If ExtractIconEx(FileName,IconIdx,
                        LargeIcon,SmallIcon,1)>0 Then
          picSmallIcon.AutoRedraw=True
          picLargeIcon.AutoRedraw=True
          'Disegna l'icona
            DrawIconEx picSmallIcon.hDC,1,1,
                         SmallIcon, 0,0,0,0,DI_NORMAL
            DrawIconEx picLargeIcon.hDC,1,1,
                          LargeIcon,0,0,0,0,DI_NORMAL
            DestroyIcon SmallIcon
            DestroyIcon LargeIcon
            picSmallIcon.Refresh
            picSmallIcon.AutoRedraw=False
            picLargeIcon.Refresh
```

I CONNECTING FILE

A partire da Windows 2000 esiste un'interessante particolarità legata alle operazioni che si possono compiere su file e directory ossia esiste la possibilità di collegare ad un file HTML un insieme di file contenuti all'interno di una directory. Così facendo, ogni qualvolta viene effettuata un'operazione di copia o spostamento sul file HTML, tutta la directory ad esso collegato (e quindi anche i file in essa contenuti) subiranno lo stesso trattamento. Per poter sfruttare questa funzionalità, è sufficiente seguire pochi semplici passi:

- **Posizionarsi nella directory di prova (ad esempio Test);**
- **Creare al suo interno un file con estensione htm o html (ad esempio Pippo.htm);**
- **Creare una sottodirectory di nome uguale al file HTML, ma senza estensione e seguito dal suffisso _files (ad esempio Pippo_files);**
- **Copiare al suo interno i file che vogliamo connettere.**

A questo punto, ogni operazione su Pippo.htm si rifletterà anche sui file contenuti in *Test\Pippo_files*. Questa opzione è abilitata per default e può essere disabilitata attraverso registry).
Nel caso sia attiva, ma si volesse operare con la funzione SHFileOperation() solo su alcuni dei file collegati, è possibile servirsi del flag *FOF_NO_CONNECTED_ELEMENTS.*

```
            picLargeIcon.AutoRedraw=False
         End If
    End If
End Sub
```

INFORMAZIONI SUI FILE

Esistono diverse funzioni undocumented per ottenere informazioni dai file. In particolare:

- **SHGetExtension:** ritorna l'estensione del file;
- **SHGetFileName:** ritorna il nome del file privo del percorso;
- **SHPathIsRelative:** ritorna un valore non nullo se il primo carattere è diverso da "\" (non è un percorso UNC) o il secondo carattere è diverso da ":";
- **SHPathIsExe:** ritorna True se il file ha l'estensione di un file eseguibile (.bat, exe, .com e .pif);
- **SHFileExists:** ritorna un valore non nullo se il percorso passato come parametro è un percorso UNC valido;
- **SHGetPathArgs:** ritorna la stringa che inizia dopo il primo spazio trovato all'interno del file specificato;
- **SHGetShortPathName:** ritorna il percorso del file in formato "short" (ossia 8.3).

Vediamo un semplice esempio che le utilizza tutte

```
Private Sub File1_Click()
   Dim sTB As String
   'Normalizza...
     txtPaths=NormalizzaPercorso(File1.Path)&File1
   sTB=txtPaths
  'Visualizza le informazioni relative alla selezione fatta
    lblGetExtension="GetExtension:
                                         "&GetExtension(sTB)
    lblGetFilename="GetFileName:  "&GetFileName(sTB)
```

UN PROGRAMMA PER RESETTARE IL COMPUTER

INIZIALIZZIAMO IL PROGRAMMA

```
   'Impostiamo a 0 il flag
     BIFoption=0
   'Mostriamo all'avvio il pannello 0
     Picture1(0).ZOrder 0
   'Carichiamo la lista degli special folder
     With cmbSpecialFolder
      .AddItem "CSIDL_DESKTOP {desktop}"
      .ItemData(.NewIndex)=&H0
      .AddItem "CSIDL_INTERNET Internet Explorer
                                   (icon on desktop)"
      .ItemData(.NewIndex)=&H1
      .AddItem "CSIDL_PROGRAMS Start Menu\Programs"
      .ItemData(.NewIndex)=&H2
ECC...
   End With
   'Se il sistema operativo è Windows NT-like...
     If IsWinNT Then
        TxtRestartPrompt="ioProgrammo Restart Program" +
                                        vbCrLf + vbCrLf
        With cmbRestartOp
             .AddItem "0 - EWX_LOGOFF"
             .AddItem "1 - EWX_SHUTDOWN"
             .AddItem "2 - EWX_REBOOT"
             .AddItem "4 - EWX_FORCE"
             .AddItem "8 - EWX_POWEROFF"
        End With
     Else
        TxtRestartPrompt="Attenzione...!" & vbCrLf & _
        "Se si preme Sì, Windows chiuderà tutte le sessioni ed" &
        "vbCrLf & _i programmi senza nessuna richiesta di
        "conferma!" & vbCrLf & vbCrLf
        With cmbRestartOp
             .AddItem "1 - EXIT senza prompt"
             .AddItem "2 - REBOOT system"
        End With
     End If
End Sub
```

1 **Inizializziamo le variabili necessarie ed impostiamo i valori delle listbox utilizzate nel programma.**

IMPOSTIAMO IL FILTRO DI BROWSING

```
 ' Valorizza il flag BIFOption in base al valore della proprietà
 ' TAG del controllo selezionato/deselezionato. La proprietà Tag
 ' del controllo contiene, per ogni elemento dell'array
 ' CheckBIF, il corretto valore decimale da aggiungere/sottrarre.
    If CheckBIF(Index).Value = 1 Then
       BIFoption = BIFoption + CheckBIF(Index).Tag
    Else
       BIFoption = BIFoption - CheckBIF(Index).Tag
    End If
```

2 **Impostiamo la variabile BIFoption servendoci della proprietà Tag delle checkbox.**

MOSTRIAMO LA FINESTRA DI DIALOGO

```
 ' Inizializza la struttura
    With BrwInf
         .hwndOwner = Me.hWnd
         .pszDisplayName = Space(260)
         .lpszTitle = ("Shell32 API - by ioProgrammo")
         .ulFlags = BIFoption
    End With
  IDList = SHBrowseForFolder(BrwInf)
  If IDList Then
     sPath = String$(MAX_PATH, 0)
     SHGetPathFromIDList IDList, sPath
     ' Libera IDList
       CoTaskMemFree IDList
     iNull = InStr(sPath,vbNullChar)
     If iNull Then
        sPath = Left$(sPath, iNull - 1)
     End If
  End If
  ' Ritorna le informazioni
    txtPath.Text = sPath
    txtSelectedFolder.Text = BrwInf.pszDisplayName
    txtIdxImage.Text = (BrwInf.iImage)
```

3 **Mostriamo la finestra di dialogo filtrata e la selezione dell'utente.**

```
   lblIsPathRelative="IsPathRelative:
                                   "&IsPathRelative(sTB)
   lblIsPathExe="IsPathExe:          "&IsPathExe(sTB)
   lblFileExists="FileExists:         "&FileExists(sTB)
   lblGetArgs="GetArgs:             "&GetArgs(sTB)
   lblGetShortPath="GetShortPath:   "&GetShortPath(sTB)
End Sub
```

La funzione NormalizzaPercorso() non fa altro che "normalizzare" la stringa passata come parametro (ossia il percorso di un file), aggiungendo, se occorre, un "\" al termine.

LA FUNZIONE SHFILEOPERATION()

Una delle funzioni più interessanti della libreria Shell32, è la SHFileOperation(). Attraverso essa è possibile compiere operazioni come copia e cancellazione di file demandando la maggior parte dei compiti "gravosi" al sistema operativo. Non fa parte delle undocumented function di Shell32, ma si è pensato che poteva comunque essere interessante dare un'occhiata alle sue "capacità".
La sintassi della funzione è la seguente:

```
Declare Function SHFileOperation Lib "Shell32.dll" Alias
     "SHFileOperationA" (lpFileOp As SHFILEOPSTRUCT)
                                               As Long
```

La funzione ritorna zero in caso di successo. Il parametro lpFileOp deve riferirsi ad una struttura "valida" ed il compito di valorizzarla, ovviamente, spetta al programmatore. Nel caso essa punti ad una struttura nulla o con item "errati", è possibile che si verifichino risultati imprecisabili. Esistono alcuni "piccoli" dettagli circa l'utilizzo della funzione *SHFileOperation*() ed uno di questi, molto importante, è quello che riguarda l'operazione d'eliminazione dei file. Va ricordato, infatti che se il valore di fFlags non "contiene" l'opzione FOF_ALLOWNDO, un'operazione siffatta porta alla cancellazione permanente del file o dei file oggetto dell'operazione, ossia senza che questi passano attraverso il Cestino. Cominciamo subito col vedere un esempio sulla copia dei file:

```
Function SHCopiaFile() As Long
  'Inizializza la struttura
    With SHFileOp
         .hWnd=frmPrincipale.hWnd
         .wFunc=FO_COPY
         .pFrom=FileOrigine&String$(2,0)
         .pTo=Destinazione&"\"&String$(2,0)
         .fFlags=FOF_SIMPLEPROGRESS
         '.lpszProgressTitle="Operazione in corso...
                                      "&String$(2,0)
   End With
   SHCopiaFile=SHFileOperation(SHFileOp)
    'GESTIONE ERRORI...
End Function
```

Tutto quello che occorre fare è compilare opportunamente la struttura SHFileOp (rammentando di valorizzare nella maniera opportuna l'item wFunc e di terminare con un doppio zero le stringhe pFrom e pTo) e richiamare la funzione passandole proprio quest'oggetto. In questa funzione, così come nelle restanti, non sono stati gestiti gli errori solo per non diminuire la leggibilità del codice, ma chiunque può migliorare questa e le restanti completando questo "pezzo mancante". Per maggiore chiarezza, mostriamo la funzione che si occupa di gestire l'eliminazione di un file:

Fig. 2: *Le API per ottenere informazioni sui file in azione*

```
Public Function SHEliminaFile() As Long
  'Inizializza la struttura
  '(l'item pTo può essere ignorato)
    With SHFileOp
         .hWnd=frmPrincipale.hWnd
         .wFunc=FO_DELETE
         .pFrom=FileOrigine&String$(2,0)
         .fFlags=FOF_ALLOWUNDO Or FOF_
                                    SIMPLEPROGRESS
         '.lpszProgressTitle="Operazione in corso...
                                     "&String$(2,0)
    End With
    SHEliminaFile=SHFileOperation(SHFileOp)
    'GESTIONE ERRORI...
End Function
```

In questa funzione, l'item pTo può essere completamente ignorato poiché non risulta necessario per operazioni di cancellazione.
Naturalmente, fFlags, per maggiore sicurezza, va impostato con il flag FOF_ALLOWUNDO per consentire il recupero del file direttamente dal Cestino di Windows.

CONCLUSIONI

Come abbiamo potuto constatare, la libreria *Shell32 .dll* ci consente di effettuare moltissime operazioni, più o meno semplici, che potrebbero tornarci in molti casi quando le normali funzioni disponibili non ci aiutano.

Francesco Smelzo

Tecnologia XML e database Access

XML e Visual Basic da campionato!

Completiamo l'applicazione che gestisce il calendario di un torneo a squadre con la presentazione dell'algoritmo e la descrizione delle routine che interrogano un documento e un database in formato XML

Discussione aperta all'indirizzo
http://forum.ioprogrammo.net/thread.php?threadid=4047&boardid=13

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
Conoscenze di base sulla tecnologia XML, sui controlli MSHFlexGrid, WebBrowser e ADODB

Software
Windows 98 o sup. Visual Basic 6 SP6

Impegno

Tempo di realizzazione

Questo è il secondo e ultimo appuntamento dedicato all'applicazione che gestisce il calendario di un campionato di calcio all'italiana; nel precedente appuntamento sono stati presentati vari aspetti funzionali, la struttura dati, le parti principali dell'interfaccia ed è stato fatto un breve cenno all'algoritmo di ottimizzazione utilizzato. Si è visto che i dati principali delle squadre di calcio sono archiviati in un database Access e che la struttura dati di supporto all'applicazione è un documento XML. Inoltre, sono stati presentati diversi argomenti correlati alla struttura dati utilizzata, quali File e alberi XML, Parser MSXML e oggetti DOM (Document Object Model) - il modello ad oggetti per manipolare un documento XML. Sono state fatte, anche, diverse considerazioni sul calcolo combinatorio e sulla notazione XPath (XML Path Language) usata per interrogare i documento XML. Si ricorda che per i nostri esemi abbiamo usato un database calcio.mdb contenuto nel codice allegato alla rivista, include una sola tabella nominata squadra ed ha i seguenti campi: *codiceSQ*, *nomeSQ*, *calendario* e *Note*. Il form principale dell'applicazione contiene diversi pulsanti, una MSHFlexGrid e un WebBrowser. I pulsanti abilitati, nel precedente appuntamento, sono *creaalberoxml* e *salvaalbero*. Le routine associate a questi due pulsanti permettono di "inizializzare" l'albero XML del campionato, di visualizzarlo nell'oggetto WebBrowser e di salvarlo in un file XML e nel campo calendario della tabella Squadre. In questo appuntamento completeremo l'applicazione implementando le routine che permettono di generare il calendario e visualizzarlo.

codiceSQ	nomeSQ	Gio. 1	Gio. 2	Gio. 3	Gio. 4	Gio. 5
1	Inter	Juventus	Palermo	Milan	Napoli	Messina
2	Milan	Messina	Napoli	Inter	Juventus	Palermo
3	Juventus	Inter	Messina	Palermo	Milan	Napoli
4	Napoli	Palermo	Milan	Messina	Inter	Juventus
5	Palermo	Napoli	Inter	Juventus	Messina	Milan
6	Messina	Milan	Juventus	Napoli	Palermo	Inter

Fig. 1: ***Il form principale mostra i dati di un campionato a 6 squadre***

VINCOLI DA RISPETTARE

Alla base dell'applicazione, c'è un algoritmo di ottimizzazione combinatoria che considerate **n** squadre trova gli accoppiamenti che soddisfano determinati vincoli. In particolare per l'applicazione presentata in quest'articolo sono considerati i seguenti vincoli:

1. una squadra non può incontrare se stessa;
2. due squadre possono incontrarsi se non si sono incontrate nei turni precedenti;
3. le squadre che partecipano al campionato devono essere almeno 4 e in ogni caso non possono essere in numero dispari.

Sulla base di ciò i turni di campionato sono n-1 e le partite per turno sono n/2. Si ricorda che il documento XML del campionato è strutturato con i seguenti i tag: *<campionato>*, *<squadre>*, *<squadra>*, *<codiceSQ>*, *<nomeSQ>* e *<partite>*. *Campionato* è il tag radice, squadre ha come nodi figli squadra che a sua volta ha come figli codiceSQ, nomeSQ e partite. Quest'ultimo è strutturato con i tag: *<giornata>* e *<SQavversaria>*.

L'ALGORITMO

Dopo queste premesse si descrive il codice che permette di riempire i rami partite per ogni giornata. Il processo di creazione dell'albero XML del campionato può essere schematizzato nel modo seguente:

- si parte considerando l'albero XML inizializzato con i dati delle squadre (letti dal database);

- il nodo **n**, di quest'albero, è copiato in una variabile di comodo per essere utilizzato come elemento riempitivo in base al numero - pari o dispari - della giornata;
- i primi **n-1** nodi sono copiati e reinseriti nell'albero a partire dalla posizione n, poi è inserito il nodo di comodo.

Dopo i passaggi precedenti, l'albero avrà *(n-1)+(n-1)+1=2n-1* nodi.

- Sull'albero, allungato, per ogni giornata, con due cicli i cui estremi e step sono definiti a hoc, s'inseriscono gli incontri nei nodi partite;
- dopo i due cicli, sempre per ogni giornata, in partite è inserito l'elemento salvato nella variabile di comodo;
- infine, dopo aver elaborato tutte le giornate, sono eliminati gli n-1 nodi inseriti e l'albero ritorna ad essere di n nodi.

I passi precedenti sono implementati con la procedura *GeneraCalendario*, invocata con il pulsante *Genera* del form principale. La *GeneraCalendario* dopo aver creato il calendario visualizza il contenuto dei nodi partite nella *MSHFlexGrid*. Il calendario completo del campionato, invece, si può ricavare con il form *FrmSelezione*, non presentato nell'articolo ma inserito nel progetto fornito con il CD allegato alla rivista. Form *FrmSelezione* si avvia con il pulsante Interroga del form principale.

CREARE IL CALENDARIO

In questo paragrafo è introdotta la procedura GeneraCalendario - che crea il calendario XML - e le procedure correlate, vale a dire:

- Aggiunginodi e Cancellanodi, che permettono di aggiungere ed eliminare dei nodi dall'albero XML;
- InserisciIncontro e InserisciIncontroTemp, che consentono d'inserire gli avversari nei rami partite.

In particolare la *InserisciIncontroTemp* è utilizzata per inserire la squadra di posizione n, cioè quella salvata nella variabile di comodo. Questa variabile è nominata *SQComodo* ed insieme con altri elementi è dichiarata in un modulo come mostrato di seguito:

```
Public n As Integer
Public xmldocument As DOMDocument
Public Type Squadra
 nome As String
 codice As Integer
End Type
Public SQComodo As Squadra
```

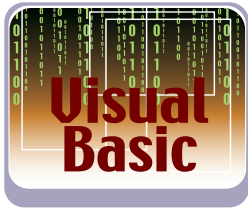
Si fa notare che n contiene il numero di squadre partecipanti al campionato e che *SQComodo* è di un tipo che raggruppa un nome e un codice. La variabile **n** è impostata quando si leggono i nomi delle squadre dal database. La procedura *GeneraCalendario* è la seguente.

```
Private Sub GeneraCalendario_Click()
 If n = 0 Then
  MsgBox "Prima Caricare le squadre", vbCritical
  Exit Sub
 End If
`bisogna mischiare i nodi
 Dim comlist As IXMLDOMNodeList
 Dim num As Integer
 Set comlist = xmldocument.documentElement
                    .selectNodes("//squadre/*")
 num = comlist.length
 SQComodo.codice = comlist.Item(num - 1)
                    .childNodes.Item(0).Text
 SQComodo.nome = comlist.Item(
                    num - 1).childNodes.Item(1).Text
 n = n - 1
 aggiunginodi
 n2 = (n - 1) / 2
 For turno = 1 To n
  For pos = turno + 1 To turno + n2 Step 2
   InserisciIncontro pos, turno + turno - pos + n, turno
  Next pos
  For pos = turno + 2 To turno + n2 Step 2
   InserisciIncontro turno + turno - pos + n, pos, turno
  Next pos
  Dim codice As Integer
  Dim nome As String
  codice = SQComodo.codice
  nome = SQComodo.nome
  If ((turno Mod 2) <> 0) Then
   InserisciIncontrotemp turno, 1
  Else
   InserisciIncontrotemp turno, 2
  End If
 Next turno
 Cancellanodi
 Caricagriglia
 Salva.Enabled = True
 `abilita il pulsante salva
 GeneraCalendario.Enabled = False
`disabilita il pulsante
End Sub
```

Nella *GeneraCalendario* dopo aver impostato l'ultimo nodo nella variabile *SQComodo* e decrementato **n** di un'unità (dato che diminuiscono le squadre) è invocata la procedura *AggiungiNodi* che allunga l'albero.

```
Private Sub AggiungiNodi()
 Dim figlio As IXMLDOMElement
```

SMART DOCUMENT

Gli *Smart Document* sono strumenti di Office 2003 particolarmente utili per la creazione di soluzioni che si sviluppano secondo un determinato processo. Gli *Smart Document* possono essere implementati, con vari linguaggi (Visual Basic 6, C#, C++ e VB Net) sia in tecnologia COM che .NET e possono interagire con database Access e SqlServer. Gli *Smart Document* possono essere sviluppati per una rete locale o geografica e quindi usare Web Service e Siti Web basati su *SharePoint Portal Server 2003.* Gli *Smart Document* sono sviluppati con *Smart Document SDK.* Sarebbe interessante sviluppare uno *Smart Document* per la gestione dei calendari!

```
Dim Qpadre As IXMLDOMElement
Dim i As Integer
For i = 1 To n
 Set Qpadre = xmldocument.createNode(
                              NODE_ELEMENT, "squadra", "")
 xmldocument.selectSingleNode(radice
                         + "/squadre").appendChild Qpadre
 Set figlio = xmldocument.createNode(
                                 "element", "codiceSQ", "")
 Qpadre.appendChild figlio
 Set figlio = xmldocument.createNode(
                                   "element", "nomeSQ", "")
 Qpadre.appendChild figlio
 Set figlio = xmldocument.createNode(
                                  "element", "partite", "")
 Qpadre.appendChild figlio
 Set figlio = Nothing
Next i
Dim comlist As IXMLDOMNodeList
Set comlist = xmldocument.documentElement
                                .selectNodes("//squadre/*")
For i = 1 To n
 comlist.Item(i + n - 1).childNodes.Item(0).Text = _
              comlist.Item(i - 1).childNodes.Item(0).Text
 comlist.Item(i + n - 1).childNodes.Item(1).Text = _
              comlist.Item(i - 1).childNodes.Item(1).Text
Next i
comlist.Item(n + n).childNodes.Item(0).Text =
                                        SQComodo.codice
comlist.Item(n + n).childNodes.Item(1).Text =
                                          SQComodo.nome
End Sub
```

L'AggiungiNodi clona **n** nodi (con n = numero squadre-1) utilizzando due cicli. Con il primo inserisce n nodi squadra ed i relativi nodi figli (*codiceSQ*, *nomeSQ* e *partite*); col secondo, invece, copia i valori dei primi n nodi in quelli inseriti. Dopo i due cicli, infine, si copiano i valori della variabile *SQComodo* nell'ultima posizione dell'albero. Si fa notare che la numerazione dei nodi nella collezione *comlist* inizia da zero. Per esempio quando il numero di squadre n=6, dopo queste operazioni l'albero XML ha 11 nodi vale a dire 2 volte i primi 5 nodi più il nodo *SQComodo*. Dato che i nodi aggiunti con la procedura precedente sono soltanto di supporto, alla fase di creazione del calendario, devono essere eliminati alla fine del processo. Questo si fa attraverso la seguente procedura.

```
Private Sub Cancellanodi()
Dim com As IXMLDOMNode
Dim comlist As IXMLDOMNodeList
Set com = xmldocument.documentElement
                           .selectSingleNode("//squadre")
Set comlist = xmldocument.documentElement
                                .selectNodes("//squadre/*")
For i = n To comlist.length - 2
  com.removeChild comlist.Item(i)
Next
End Sub
```

Nella *Cancellanodi* prima è valutato il numero complessivo di nodi dell'albero - utilizzando la proprietà *length* della collezione di nodi - e poi, in un ciclo, sono eliminati i nodi al disotto del nodo n, tranne l'ultimo.
Si noti che –2 è necessario dato che i nodi sono numerati a partire da zero e l'ultimo nodo dell'albero non deve essere cancellato (dato che contiene i dati della squadra di "comodo").
Le procedure *InserisciIncontro* e *InserisciIncontroTemp* sono le seguenti.

```
Public Sub InserisciIncontro(pos1 As Integer, pos2
                       As Integer, _ giornata As Integer)
 Dim comlist As IXMLDOMNodeList
 Dim ele1 As Squadra
 Dim ele2 As Squadra
 Set comlist = xmldocument.documentElement
                                .selectNodes("//squadre/*")
 ele1.codice = comlist.Item(pos1).childNodes.Item(0).Text
 ele1.nome = comlist.Item(pos1).childNodes.Item(1).Text
 ele2.codice = comlist.Item(pos2).childNodes.Item(0).Text
 ele2.nome = comlist.Item(pos2).childNodes.Item(1).Text
 AppendIncontro ele1, ele2, giornata
End Sub
```

La *InserisciIncontro* è invocata dalla *GeneraCalendario* quando è necessario inserire un incontro tra due squadre, per questo i parametri della procedura sono due posizioni (pos1 e pos2) e il turno. In base a questi parametri dall'albero vengono ricavati il nome e il codice delle squadre e inseriti, per il turno specificato, una nel ramo partite dell'altra. Per inserire gli incontri della squadra di comodo, invece, è utilizzata la seguente.

```
Public Sub InserisciIncontrotemp(giornata As Integer,
                                  qualepara As Integer)
 Dim comlist As IXMLDOMNodeList
 Dim pos As Integer
 pos = giornata
 Dim ele1 As Squadra
 Dim ele2 As Squadra
 Set comlist = xmldocument.documentElement
                                .selectNodes("//squadre/*")
 If qualepara = 1 Then
  ele1 = SQComodo
  ele2.codice = comlist.Item(pos).childNodes.Item(0).Text
  ele2.nome = comlist.Item(pos).childNodes.Item(1).Text
 Else
  ele2 = SQComodo
  ele1.codice = comlist.Item(pos).childNodes.Item(0).Text
  ele1.nome = comlist.Item(pos).childNodes.Item(1).Text
 End If
```

```
 AppendIncontro ele1, ele2, giornata
End Sub
```

La *InserisciIncontriTemp* è invocata dalla *GeneraCalendario* in base al tipo di giornata, dispari o pari; questo giustifica i tipi di parametri. Le due procedure precedenti per aggiornare i nodi utilizzano *l'AppendIncontri*, mostrata in seguito.

```
Public Sub AppendIncontro(ele1 As Squadra, _
                       ele2 As Squadra, giornata As Integer)
 Dim com As IXMLDOMElement
 Set com = xmldocument.createNode(NODE_ELEMENT,
                                              "giornata", "")
 com.Text = CStr(giornata)
 xmldocument.selectSingleNode("campionato
   /squadre/squadra [ codiceSQ = " & CStr(ele1.codice)
                              + " ]/partite").appendChild com
 Set com = xmldocument.createNode(NODE_ELEMENT,
                                       "SQavversaria", "")
 com.Text = CStr(ele2.nome)
 xmldocument.selectSingleNode("campionato
   /squadre/squadra [ codiceSQ = " & CStr(ele1.codice)
                              + " ]/partite").appendChild com
 Set com = xmldocument.createNode(NODE_ELEMENT,
                                              "giornata", "")
 com.Text = CStr(giornata)
 xmldocument.selectSingleNode("campionato
   /squadre/squadra [ codiceSQ = " & CStr(ele2.codice)
                              + " ]/partite").appendChild com
 Set com = xmldocument.createNode(NODE_ELEMENT,
                                       "SQavversaria", "")
 com.Text = CStr(ele1.nome)
 xmldocument.selectSingleNode("campionato
   /squadre/squadra [ codiceSQ = " & CStr(ele2.codice)
                             + " ] /partite").appendChild com
End Sub
```

Nel paragrafo successivo si descrive la procedura che carica gli avversari delle squadre nella MSFlexGrid della maschera principale, cioè la procedura *CaricaGriglia* invocata dalla *GeneraCalendario* dopo aver cancellato i nodi di supporto.

VISUALIZZARE LE PARTITE IN UNA GRIGLIA

In questo paragrafo è descritta la procedura CaricaGriglia che popola la MSHFlexGrid con i dati del documento XML che contiene il calendario degli incontri. Si presuppone che il documento XML sia già caricato, altrimenti si può fare seguendo le indicazioni date nel precedente articolo.

La *CaricaGriglia* legge i dati da un documento XML nominato xmldocument e l'inserisce nella MSHFlexGrid1. Questo viene fatto secondo i seguenti passi:

1. si valuta il numero di nodi squadra presenti nell'albero;
2. in base al numero di squadre s'impostano il numero di colonne e di righe della griglia;
3. con due ciclo, per ogni squadra e per ogni giornata, s'inseriscono i dati nella griglia;
4. s'imposta il numero di colonne fisse.

La procedura è la seguente.

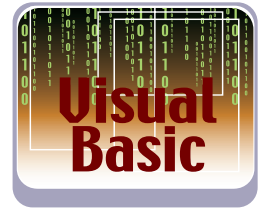
```
Sub caricagriglia()
 Dim Qlist As IXMLDOMNodeList
 Dim Plist As IXMLDOMNodeList
 Dim i, j As Integer
 Dim Qnum As Integer
 Dim Pnum As Integer
 Dim colonna As Integer
 Set Qlist = xmldocument.documentElement.selectNodes(
                                             "//squadre/*")
 Qnum = Qlist.length
 MSHFlexGrid1.Cols = Me.MSHFlexGrid1.Cols + Qnum – 1
 'numero di righe già impostate
  For i = 1 To Qnum
   colonna = 1
   If i <= Qnum - 1 Then
    MSHFlexGrid1.TextMatrix(0, i + 1) = "Gio. " + CStr(i)
   End If
  For j = 2 To (Qnum - 1) * 2 Step 2
   colonna = colonna + 1
   MSHFlexGrid1.TextMatrix(i, colonna) = Qlist.Item(
       i - 1).childNodes.Item(2).childNodes.Item(j - 1).Text
  Next
 Next
 MSHFlexGrid1.FixedCols = 2
End Sub
```

Si noti che è utilizzato il metodo *TextMatrix* che consente di selezionare e caricare una singola cella della griglia. Come caricare in una listbox i dati sugli incontri contenuti nel database Calcio.

CONCLUSIONI

Gli algoritmi proposti in questa miniserie di due articoli sono solo una delle tantissime implementazioni a cui si presta un problema di calcolo combinatorio come quello esposto.

Sul forum di ioProgrammo trovate tante e fantasiose soluzioni al problema, alcune delle quali molto matematiche.

È divertente notare come l'approccio alla programmazione coinvolce anche lo stile personale nella risoluzione dei problemi. Qui abbiamo preferito usare XML perché ci sembrava particolarmente pratico, ma se avete altre soluzioni da proporre, il forum vi attende.

Massimo Autiero

XPATH E SELEZIONE NODI CON []

La notazione *XPath (XML Path Language)* è usata per interrogare un documento XML, essa è supportato dal Parser MSXML dentro le XSLT e con gli elementi *selectNodes* e *selectSingleNode*. L'XPath è una notazione dichiarativa per questo con essa si descrivono delle relazioni gerarchiche tra nodi. Per esempio con la seguente espressione *"campionato/squadre/squadra [codiceSQ = " _ & CStr(2) + "]/partite"* è selezionato il nodo partite della squadra con *codiceSQ=2*

CARICARE UN DOCUMENTO XML CON INTERNET EXPLORER

Come caricare in una listbox i dati sugli incontri di calcio contenuti nel database Calcio.mdb

1 **Nel progetto includere un riferimento a *MSXML4.dll* (Microsoft XML, v4.0) e a *Microsoft Activex Data Object (ADO).* Sul form, invece, inserire almeno un textbox - nominato *txtcodiceSQ*, un pulsante - nominato *Carical* - e un listbox.**

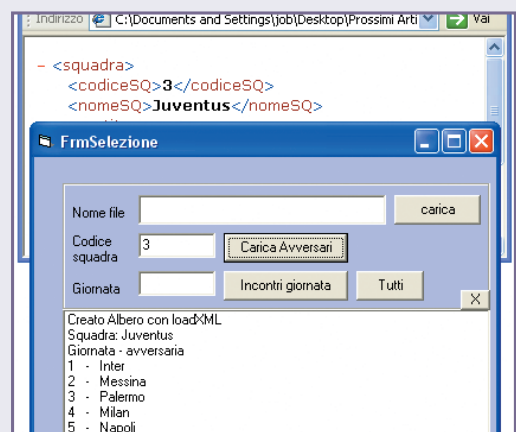
```
Private Sub Caricacal_Click()
 If Not IsNumeric(txtcodiceSQ) Then
  MsgBox "Specificare correttamente il " &
              "codice della squadra", vbCritical, "Errore"
  Exit Sub
 End If
 Dim xmldocument As DOMDocument
 Dim com As String
 Set xmldocument = trovacal("select calendario" &
       " from squadra where codiceSQ=" + txtcodiceSQ)
 Dim xmldocument2 As DOMDocument
 Set xmldocument2 = New DOMDocument
 On Error Resume Next
 com = xmldocument.selectSingleNode _
 ("//rs:data/z:row/@calendario").Text
 If com = "" Then
  Err.Clear
  MsgBox "Codice squadra errato o"
             & " database vuoto", vbCritical, "Errore"
  Exit Sub
 End If
 xmldocument2.loadXML com
 Dim comlist As IXMLDOMNodeList
 List1.AddItem "Creato Albero con loadXML"
 List1.AddItem "Squadra: " +xmldocument2
          .documentElement.childNodes.Item(1).Text
 Set comlist = xmldocument2.
        documentElement.selectNodes("//partite/*")
 List1.AddItem "Giornata - avversaria"
 For i = 0 To (comlist.length - 1) Step 2
  List1.AddItem comlist.Item(i).Text  & "   -   "
                    + comlist.Item(i + 1).Text
 Next
End Sub
```

2 **Nel pulsante devono essere inserite le istruzioni che leggono il codice della squadra da *txtcodiceSQ*, inviano la query al database e caricano i dati ricevuti in un documento XML. La procedura del pulsante è la seguente. Si fa notare che nella *Caricacal* si usano due oggetti *DOMDocument*, uno per ricevere il risultato della query e l'altro per ricostruire la struttura dell'albero XML con la *loadXML*. Si ricorda che i dati sugli incontri sono contenuti nel campo calendario riempito quando si clicca il pulsante *Salva* del form principale.**

```
Public Function trovacal(ByVal strQuery As String)
                                      As DOMDocument
 Dim xmldocument As DOMDocument
 Set xmldocument = New DOMDocument
 Dim objConn As ADODB.Connection
 Dim rs As ADODB.Recordset
 Set objConn = New ADODB.Connection
 Set rs = New ADODB.Recordset
 objConn.Open "Provider=Microsoft.Jet.OLEDB.4.0;"
 & "Data Source=" + App.Path + "\calcio.mdb", "", ""
 rs.Open strQuery, objConn
 rs.Save xmldocument, adPersistXML
 Set trovacal = xmldocument
 Set xmldocument = Nothing
 rs.Close
 Set rs = Nothing
 objConn.Close
 Set objConn = Nothing
End Function
```

3 **La *Caricacal_Click* per eseguire la query utilizza la funzione *trovacal*. Questa riceve una *query select* e restituisce un documento XML creato con il metodo *Save* del recordset.**

```
Dim strfile As String
strfile = App.Path + "\" + "calendario" _
 & xmldocument2.documentElement.childNodes. _
                             Item(1).Text + ".xml"
xmldocument2.Save (strfile)
```

4 **Infine per salvare i dati degli avversari in un file XML, alla *Caricacal* si devono aggiungere le seguenti istruzioni**

5 **Si noti che il nome del file XML è del tipo *"calendario" + "nome squadre corrente"*, dove il nome della squadra è recuperato dall'albero *xmldocument2*.**

ADODB SAVE FORMATO XML E LOADXML

Il metodo *save* di un oggetto *Recordset* permette di salvare il suo contenuto in un oggetto o in un file. La sintassi è la seguente: *recordset.Save Destinazione, PersistFormat*. *Destinazione* è il nome di un file o di un oggetto, *PersistFormat* specifica in che formato deve essere salvato il *Recordset*. *I possibili valori sono: adPersistADTG e adPersistXML*. Il primo è il formato predefinito cioè il formato binario; *AdPersistXML*, invece, è il formato XML. Il documento XML prodotto con la save presenta vari elementi quello che bisogna usare per recuperare i dati del campo calendario è "*//rs:data/z:row/@calendario*". Invece per recuperare la struttura di un documento XML da una stringa che contiene dati in formato XML si può utilizzare il metodo *loadXML* dell'oggetto *DomDocument*.

Il teorema di Bayes per difenderci dalle email indesiderate

Filtri bayesiani contro la posta spazzatura

In questo articolo parleremo dei principali sistemi di protezione, e implementeremo un filtro bayesiano creando un dizionario personalizzato, eseguendo alcuni calcoli statistici necessari

Pare che il traffico generato dalla posta spazzatura costituisca, insieme a quello prodotto dai crawler dei motori di ricerca, la maggior parte di quello prodotto sull'intera rete. In altre parole, se non ci fosse posta spazzatura e l'indicizzazione dei motori di ricerca funzionasse con modalità diverse, Internet sarebbe molto più veloce e potrebbe supportare molti più utenti con lo stesso hardware. Per proteggersi dai messaggi indesiderati sono nati diversi strumenti software. Le tecniche maggiormente utilizzate per implementare dei sistemi antispam possono riassumersi nelle seguenti categorie:

- **Ricerca di elementi chiave** - Il testo del messaggio viene analizzato e per ogni caratteristica sospetta viene assegnato un punteggio. Ad esempio, un testo solo HTML, colori forti, intestazioni di posta particolari, l'uso di termini riconducibili a posta spazzatura, sono tutti elementi che alzano il punteggio spazzatura del messaggio. Se questo supera un certo valore, il software segnala il messaggio come spazzatura.

- **white/black list** - In questo caso l'utente costruisce, in modi più o meno automatici, una lista di mittenti conosciuti ed accreditati e/o una lista di mittenti non accreditati. Il filtro di posta sui messaggi in arrivo farà passare solo i messaggi i cui mittenti sono in white list, mentre cancellerà quelli in black list. Quelli non presenti in nessuna delle due liste potrebbero essere presentati all'utente per una scelta manuale;

- **filtri statistici** - Entrambi gli approcci sopra illustrati hanno pro e contro. Ad esempio, il primo produce risultati vicini al 70%-80% ma rende quasi impossibile riconoscere il restante 30%-20% di posta. Restringendo i parametri di filtro si rischia di ottenere falsi positivi. Ovvero messaggi erroneamente indicati come posta spazzatura. L'approccio a white/black list invece presenta lo svantaggio di dover gestire manualmento l'elenco dei mittenti. Per ovviare a questi problemi è possibile utilizzare i filtri statistici. Questi sono noti anche come filtri bayesiani, dal nome dell'invetore della teoria che sta alla loro base, Thomas Bayes.

I filtri bayesiani eseguono un'analisi dei messaggi che l'utente indica essere posta spazzatura, e costruiscono un dizionario di termini "sospetti". Poi analizzano la posta accettata, e costruiscono un dizionario di termini "legittimi". Sulla base di questi dizionari, sono in grado di identificare quali messaggi potrebbero essere spazzatura o meno.
Solitamente questa analisi viene svolta all'inizio e prende il nome di "allenamento". In sostanza, appena dopo aver installato il filtro nel nostro programma di posta, è necessario passare qualche giorno indicando al sistema quale dei messaggi ricevuti è buono e quale invece è posta spazzatura. Quando la fase di allenamento viene conclusa, il sistema filtra autonomamente i messaggi, marcandoli come spazzatura o posta accettata. Il vantaggio dei filtri statistici è che la logica è basata su un teorema statistico comprovato.
Il software non azzarda valutazioni la cui logica potrebbe essere opinabile. Inoltre il filtro funziona sulla base della propria specifica fase di allenamento. Sulla base cioè della specifica posta spazzatura e posta legittima che l'utente riceve. In questo modo l'analisi è

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Linguaggio Java

Software
Java2 SDK 1.3

Impegno

Tempo di realizzazione

focalizzata e specifica. Con questi filtri si può ottenere un'approssimazione vicina al 99%.

IMPLEMENTARE UN DIZIONARIO

Il primo passo da fare per implementare un filtro bayesiano è quello di costruire un dizionario. Poi dovranno essere eseguiti alcuni calcoli statistici, ma niente di particolarmente complesso. Il dizionario contiene una serie di parole, con il numero di occorrenze delle stesse nelle mail passate per la fase di allenamento. Nei nostri esempi questa fase verrà simulata da due file mbox, uno che contiene una serie di messaggi spazzatura. L'altro contiene posta legittima.
La prima fase è dunque quella di produrre una classe che dato un file mbox ne produca il dizionario delle parole con le relative occorrenze.
Un esempio di contenuto del dizionario è:

```
Access=1.0
account=9.0
acquisition=5.0
...
commercial=3.0
communicable=0.0
communicate=1.0
Company=17.0
...
lots=1.0
Lou=5.0
love=5.0
Low=3.0
Ltd.=2.0
```

Avremo chiaramente bisogno di un dizionario delle parole corrette e uno delle parole spazzatura.

COSTRUZIONE DEL FILTRO

Una volta in possesso di dizionari ben compilati, è possibile creare il filtro di posta. Quello che verrà costruito nelle pagine seguenti è basilare, ed opera su un singolo messaggio di posta, passato come stringa o caricato da un file.
Il filtro è implementato con una singola classe *FiltroBayesiano*, dichiarata come segue:

```
public class FiltroBayesiano
{
  Dizionario postaAccettata;
  Dizionario postaRifiutata;

 /**
  * Crea un nuovo filtro, utilizzando i dizionari di posta
  * accettata e rifiutata indicati
  *
  * @param postaAccettata
  * @param postaRifiutata
  */
  public FiltroBayesiano( Dizionario postaAccettata,
                          Dizionario postaRifiutata )
 {

    this.postaAccettata = postaAccettata;
    this.postaRifiutata = postaRifiutata;
  }
}
```

il codice prevede un singolo costruttore che si aspetta i due dizionari di riferimento. Il metodo che esegue il filtro del contenuto di un file non fa altro che caricare in memoria il file:

```
public float filtra( File file ) throws IOException
{
  FileReader fr = new FileReader(file);
  char[] buffer = new char[ (int) file.length() ];
  fr.read( buffer );
  fr.close();
  return filtra( new String(buffer) );
}
```

il file viene caricato in un unico blocco di caratteri, memorizzati in un array di char. Da questo viene costruita una stringa, che viene passata al metodo filtra(String). Questo contiene la logica di valutazione del messaggio, che sostanzialmente è composta da tre passaggi:

1. estrazione di tutte le parole presenti nel messaggio, in modo similare a quanto fatto per la costruzione dei dizionari e calcolo delle probabilità dei singoli token;
2. estrazione delle parole che hanno probabilità più lontana dal punto medio, identificato da una probabilità dello 0.5.;
3. calcolo della probabilità combinata di tutti i termini, che produce la probabilità che il messaggio sia posta indesiderata.

il codice sorgente è il seguente:

```
public float filtra( String messaggio )
{
  Map tokenProb = new HashMap();
```

CLIENT DI POSTA

Alcuni client di posta che implementano direttamente un filtro bayesiano contro la posta spazzatura sono *Mail* di Apple, il programma di posta elettronica di *Mac OS X* http://www.apple.com/it/ oppure *Thunderbird*, il client di posta del progetto Mozilla http://www.mozilla.org/projects/thunderbird/

DEFINZIONE DELLA CLASSE

```
public class Dizionario
{
   public final static String TOKENS = "!#* \t\n,:";
   Map dizionario = new HashMap();
   String filename;
   /**
   * Crea un dizionario di termini e loro occorrenze
   *
   * @param filename   file che contiene la casella
                                          da analizzare
   */
   public Dizionario( String filename)
   {
      this.filename = filename;
   }
```

1 **Per prima cosa viene definita la classe Dizionario, che dispone di tre attributi: una costante che indica quali sono i separatori di parola, una Map che conterrà le parole e le occorrenze ed una stringa per ospitare il nome del file da caricare. La classe ha un costruttore che permette di specificare il nome del file da leggere**

LETTURA DEL FILE

```
public void importa() throws IOException
{
   FileReader fr = null;
   try
   {
      fr = new FileReader( filename );
      BufferedReader br = new BufferedReader( fr );
      while( true )
      {
         String line = br.readLine();
         if (line==null) {break;}
         StringTokenizer st = new
                       StringTokenizer( line, TOKENS );
         while( st.hasMoreTokens() )
         { increment( st.nextToken() );}
      }
   }
   finally
   {
      if (fr == null) {fr.close();
   }
   System.out.println("Importata la casella "
                + filename +  " per un totale di " +
                          dizionario.size() + " token");
   // System.out.println( dizionario );
}
```

2 **La seconda fase prevede la lettura del file di testo, utilizzando un oggetto FileReader associato ad un BufferedReader. In questo modo viene letta una riga alla volta e spezzata in token attraverso un oggetto StringTokenizer. Questo viene costruito con la costante TOKENS definita al passo precedente. In questo modo viene considerata una parola unica tutto quanto non separato dai caratteri sopra indicati. Per ciascuna parola così ottenuta viene chiamato il metodo increment(), che si occupa di aumentare il conteggio di questa parola, se non esiste**

INCREMENTO DEL CONTATORE

```
void increment( String token )
{
   Float count = (Float)dizionario.get(token);
   if (count == null)
   {
      count = new Float(1);
   }
   else
   {
      count = new Float( count.floatValue() + 1 );
   }
   dizionario.put( token, count );
}
```

3 **Il metodo *increment()* verifica se nel dizionario è già presente la parola. In questo caso ne ottiene il numero di occorrenze, che aumenta di uno, per poi reinserire i dati aggiornati nel dizionario. Nel caso la parola non esista, viene inserita nel dizionario con contatore a 1**

FUNZIONI DI SUPPORTO

```
float get( String token )
{
   Float value = (Float)dizionario.get(token);
   if (value == null)
  {return 0.0f;}
  else
  {
     return value.floatValue();
  }
}
```

4 **La classe viene poi integrata con due funzioni aggiuntive, che permettono di investigare i dati del dizionario da un'altra classe. Il metodo get() ritorna il numero di occorrenze di una parola passata come parametro. Se questa non esiste nel dizionario viene ritornato 0.0f. Il dato viene restituito come tipo primitivo float per agevolare le operazioni successive**

NUMERO DI PAROLE NEL DIZIONARIO

```
int size() {return dizionario.size();}
```

5 **L'altro metodo presente è *size()*, che ritorna il numero di parole nel dizionario**

COSTRUIAMO IL DIZIONARIO

```
Dizionario dizionarioJunk = new
                              Dizionario("junk.mbox");
dizionarioJunk.importa();
Dizionario dizionarioNormale = new
                          Dizionario("normale.mbox");
dizionarioNormale.importa();
```

6 **A questo punto è possibile costruire i due dizionari necessari all'operatività di un filtro bayesiano. Il codice seguente crea i due dizionari dizionarioJunk e dizionarioNormale caricando i due file junk.mbox e normale.mbox, costruiti estraendo caselle di poste create da un programma di posta elettronica.**

CREAZIONE FILE MBOX

Il programma di esempio cerca i file *junk.mbox* e *normale.mbox* dalla directory corrente. Questi file dovranno contenere un certo numero di messaggi spazzatura e legittimi, per la fase di allenamento del filtro.

CREAZIONE DEI MESSAGGI DI PROVA

I messaggi da analizzare sono contenuti nei file *bad.txt, bad1.txt, good.txt* e *good1.txt.* si noti che possono includere le intestazioni di posta o meno. I primi due sono messaggi spazzatura, gli altri due messaggi legittimi.

ALL'ENNESIMA POTENZA

Quanto mostrato in questo articolo è basilare e tratta direttamente con i meccanismi statistici di base per implementare i teoremi di Bayes. Su *Sourceforge.net* è però presente un progetto che integra ed estende queste funzionalità, una libreria di classificazione di termini che implementa anche un filtro bayesiano. È disponibile all'indirizzo http://classifier4j.sourceforge.net/

Una fonte d'informazioni preziosa per conoscere maggiori dettagli sui filtri bayesiani, tutta la teoria collegata, e molti altri argomenti, è l'enciclopedia gratuita *Wikipedia - the free encyclopedia.* Un enorme wiki su qualsiasi argomento, dispone di moltissimi articoli. I filtri bayesiani sono ottenibili all'indirizzo http://en.wikipedia.org/wiki/Bayesian_filtering

```
StringTokenizer st = new StringTokenizer(
                        messaggio, Dizionario.TOKENS );

while( st.hasMoreTokens() )
{
    String token = st.nextToken();
    tokenProb.put( token, calcolaProbabilità(token));
}

    System.out.println( tokenProb );
    float[] pb = estraiProbabilità( tokenProb );
    return probabilitàCombinata( pb );
}
```

il primo passo viene espletato tramite l'oggetto di tipo StringTokenizer. L'approccio è simile alla classe Dizionario ma nella mappa, per ogni parola, non è presente il numero di occorrenze, ma la probabilità calcolata con il metodo *calcolaProbabilità()*. Questo metodo mette in relazione le occorrenze della parola nei dizionari di posta accettata e rifiutata, calcolando la probabilità che la mail che contiene questa parola sia indesiderata. Se una parola non è presente nei dizionari assume un valore di 0.4:

```
Float calcolaProbabilità( String token )
{
  float result = 0.4f;
  float g = postaAccettata.get(token) * 2;
  float b = postaRifiutata.get(token);
  if ( !((g+b) < 5) )
  {
    float temp1 = b / postaRifiutata.size();
    temp1 = Math.min( temp1, 1 );

    float temp2 = g / postaAccettata.size();
    temp2 = Math.min( temp2, 1 );

    float temp = temp1 / (temp1+temp2);

    result = Math.min( 0.99f, temp );
    result = Math.max( 0.1f, result );
  }
  return new Float(result);
}
```

la formula utilizzata non è altro che quella descritta nella documentazione di Graham (si veda il box "Un piano per la posta indesiderata").

CALCOLI STATISTICI

Una volta ottenuto il dizionario dei token del messaggio da valutare, vengono considerati i 15 elementi con maggior discostamento dal punto medio 0.5. Si noti che il discostamento deve considerare il segno.
Ad esempio, un valore 0.8 si discosta di 0.3 dal punto medio, mentre un valore 0.3 si discosta di 0.2:

```
| 0.8 – 0.5 | = 0.3
| 0.3 – 0.5 | = 0.2
```

per ottenere l'elenco dei token ordinati per discostamento, viene costruita una mappa ordinata. Nella mappa la chiave è il punteggio, mentre il valore è la probabilità. Da questa mappa vengono estratti i primi quindici elementi e ritornati sottoforma di array di float:

```
float[] estraiProbabilità( Map dizionario )
{
  final int size = 15;
  float[] pb = new float[ size ];

  //crea una mappa con le probabilità ed i punteggi
  //relativi cioè la distanza dal valore medio
  Map punteggi = new TreeMap();
  for( Iterator iter = dizionario.keySet().iterator();
  iter.hasNext(); )
  {

    String key = (String)iter.next();
    Float currentObject = (Float)dizionario.get(key);
    float current = currentObject.floatValue();
    float punteggio = Math.abs( current - 0.5f );

    punteggi.put( new Float( punteggio ),
                              currentObject );
  }

  //la mappa è ordinata per chiave, dunque estraendo i
  //primi 15 elemeni si anno le probabilità più lontane
  //dal valore medio
  int i = 0;
  for( Iterator iter = punteggi.keySet().iterator();
                              iter.hasNext(); )
  {
     Float key = (Float)iter.next();
     Float currentObject = (Float)punteggi.get(key);

     pb[ i++ ] = currentObject.floatValue();
     if (i>=size)
     {
        break;
     }
  }

  return pb;
}
```

una volta in possesso di questo array di probabilità, è possibile calcolare la probabilità combinata. Cioè la probabilità che tutti i casi si verifichino.
La formula relativa è:

abc
abc + (1-a)(1-b)1-c)

dove a, b e c sono valori di probabilità.
Ovviamente la formula si espande fino a contemplare tutte e quindici le probabilità estratte al passo precedente. La probabilità combinata è implementata nel metodo seguente:

```
float probabilitàCombinata( float[] pb )
{
  float result = 0;

  //calcola il prodotto delle probabilità
  float prodotto = pb[0];
  for( int i=1; i<pb.length; i++ )
   {
     prodotto *= pb[i];
   }

  //calcola il prodotto di 1-x
  result = (1 - pb[0]);
  for( int i=1; i<pb.length; i++ )
   {
     result *= (1 - pb[i]);
   }

  return prodotto / ( prodotto + result );
}
```

si noti che l'array di float passato come parametro potrebbe essere di qualsiasi lunghezza. Per prima cosa viene calcolato il prodotto delle probabilità, in seguito viene calcolato il prodotto di tutte le probabilità 1-x.

IN ESECUZIONE

Il valore ritornato dal metodo filtra() è un valore float che può assumere un valore vicino allo zero. In questo caso il messaggio è accettato. Se il valore è invece superiore a 0.9, il messaggio è molto probabilmente posta indesiderata.
Nella classe *FiltroBayesiano* sono presenti metodi di utilità per eseguire queste valutazioni e stampare un messaggio a video. Il metodo main() contiene del codice di prova, che esegue il filtro a fronte di messaggi spazzatura (*bad*) o legittimi (*good*):

```
public static void main(String[] args) throws IOException
{
  Dizionario dizionarioJunk = new
                          Dizionario("junk.mbox");
  dizionarioJunk.importa();

  Dizionario dizionarioNormale = new
                          Dizionario("normale.mbox");
  dizionarioNormale.importa();

  FiltroBayesiano filtro = new FiltroBayesiano(
                 dizionarioNormale, dizionarioJunk);

  filtro.stampaEsitoFile("bad.txt");
  filtro.stampaEsitoFile("bad1.txt");
  filtro.stampaEsitoFile("good.txt");
  filtro.stampaEsitoFile("good1.txt");
}
```

l'output che si ottiene è simile al seguente:

Importata la casella junk.mbox per un totale di 11481 token

Importata la casella normale.mbox per un totale di 58850 token

>>> Il messaggio è posta spazzatura con una probabilità di 0.99949765

>>> Il messaggio è posta spazzatura con una probabilità di 0.9999999

>>> Il messaggio è posta accettata con una probabilità di 0.0

>>> Il messaggio è posta accettata con una probabilità di 0.0

come si nota, i messaggi sono stati correttamente identificati. In alcuni casi con una probabilità dello 0.99 periodico. In realtà, la soglia dello 0.9 è relativa, perché con questa formula si ottiene quasi sempre un valore zero o molto vicino all'uno.

Massimiliano Bigatti

ESECUZIONE

Per eseguire il programma lanciare la classe *FiltroBayesiano*, dalla directory che contiene il codice digitare, da console:

```
java –cp . it.bigatti
.antispam.FiltroBayesiano
```

CHI È PAUL GRAHAM?

Gli algoritmi descritti in questo articolo sono una implementazione adattata di quanto discusso nell'articolo di Paul Graham, "A plan for spam" (http://www.paulgraham.com/spam.html). Questa pubblicazione è stata la prima a portare in evidenza la possibilità di utilizzare il teorema di Bayes per gestire il problema della posta spazzatura. Le idee di Graham sono state poi raffinate da lui stesso in successive pubblicazioni e molto probabilmente da chi ha implementato via software queste teorie.

Come difendersi dagli attacchi dei pirati

Hacker semaforo rosso

È necessario utilizzare tutte le possibili accortezze per prevenire attacchi di tipo Denial of Service, considerando che un servizio non attivo fa perdere guadagni e, a lungo andare, credibilità

Utilizza questo spazio per le tue annotazioni

La natura "pubblica" di un server Web lo rende potenzialmente vulnerabile ed esposto a diversi tipi di attacchi. In particolare è possibile che un utente malintenzionato possa [1]:

- accedere a risorse per cui non è autorizzato (e impadronirsi di informazioni riservate o manipolare dati visibili ad altri)
- utilizzare il server Web in modo improprio (per esempio utilizzarlo per perpetrare ulteriori attacchi "nascondendo" l'origine dell'attacco, utilizzare le risorse del server per memorizzare dati non autorizzati e impropri)
- effettuare attacchi che portano ad un *Denial of Service*.

Ovviamente non sono menzionati attacchi "fisici", come il rubare l'hardware o avere accesso diretto alla console del server. Accanto agli attacchi di un utente è possibile che il server Web venga in contatto con Virus o Worms; anch'essi possono provocare danni al sistema e produrre gli stessi effetti di attacchi *Denial Of Service*. In ogni caso è sempre più vero che ad agire non è un utente ma una serie di tool automatici comandati (o programmati) da utenti malintenzionati. È comune trovare su internet tool e strumenti di test che possono essere, con poca difficoltà, adattati per perpetrare attacchi a sistemi remoti. Ma cosa si intende con *Denial of Service*?

DENIAL OF SERVICE

Con il termine *Denial of Service* (o semplicemente DoS) si identificano una serie di azioni volte a rendere uno o più servizi non disponibili ad un gruppo di utenti autorizzati per un certo periodo di tempo; se il focus si concentra su un servizio reso disponibile in rete, si può anche parlare di *Network Denial of Service;* lo scopo di tali azioni è provocare un consumo (o un comportamento) anomalo di un insieme di servizi tanto da renderli inutilizzabili [2]. Quando l'attacco avviene da più di una sorgente di parla anche di *Distributed DoS (DDoS)*.

Caso 1: una falla di SQL Server permetteva ad un attaccante di inviare una particolare sequenza di byte e di eseguire, sull'istanza del SQL Server attaccato, del codice a suo piacimento. Un warm sfruttava questa falla per attaccare un sito e far partire da esso nuovi attacchi ad indirizzi generati in maniera casuale. Questo worm ha causato enormi danni, sia perché ha generato problemi DoS sui siti attaccati sia perché ha generato un altissimo volume di traffico su Internet, rallentandone notevolmente le normali operazioni. L'unico modo per proteggere questo tipo di attacchi è quello di installare le patch per la versione di SQL che è vulnerabile oppure "nascondere" tale server dietro un firewall e renderlo inaccessibile dall'esterno.

Caso 2: con lo stesso principio, ovvero sfruttando una falla di sicurezza, un warm può attaccare numerosi siti. Ma anziché compromettere il loro funzionamento, si limita a far eseguire da essi una serie di attacchi verso un unico sito, che è il vero obiettivo dell'attacco (attacco di tipo DDoS). In questo modo il sito attaccato consuma rapidamente le sue risorse. In questo caso un semplice firewall non basta, in quanto non riesce a distinguere le richieste provenienti da utenti "normali" e richieste da siti compromessi dal worm. L'unico modo è rendere non attaccabili i siti ove il worm si potrebbe propagare.

DIVERSI TIPI DI DOS

Secondo lo stack OSI, si possono distinguere sette livelli concettuali, da quello fisico a quello applicativo. I protocolli di Internet si possono collocare a di-

versi livelli (**Figura 1**, adattata da [**2**]) e possiamo distinguere, in base ad essi, diversi tipi di attacchi DoS in base al livello OSI interessato. Resta inteso che attacchi DoS ad uno specifico livello compromettono tutti i livelli successivi (più alti): per questo motivo la maggior parte delle tecniche per aumentare la sicurezza e difendersi da attacchi DoS si è focalizzata sugli strati bassi, tralasciando spesso la sicurezza relativa agli strati alti (e in particolare per quanto concerne il livello applicativo). Questo articolo, per contro, offre esempi e possibili soluzioni per il livello applicativo.

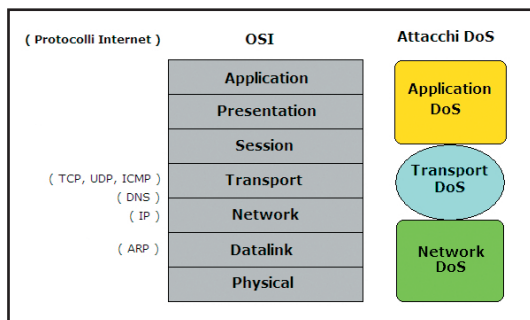



Fig. 1 : Il modello OSI e i diversi tipi di attacchi DoS

PROBLEMI APERTI

Mentre è comune che un'azienda si prepari ad affrontare attacchi DoS di basso livello, attrezzandosi con firewalls, routers e configurando in maniera opportuna application servers, spesso viene trascurata la difesa da questi tipi di attacchi a livello applicativo. Tra le cause di questa mancanza possiamo ricordare [**4**]:

- Mancanza di standard aziendali per la sicurezza nelle applicazioni sviluppate
- Carente cultura degli sviluppatori software relativamente alla sicurezza; questo porta a considerare l'uso di protocolli o infrastrutture "sicure" (SSL, https e così via) sufficiente per garantire la sicurezza delle applicazioni; oppure si genera la falsa sicurezza nell'uso di protocolli (proprietari) non documentati
- Eccessiva attenzione al *"time to market"* che porta a privilegiare il funzionamento "di base" dell'applicazione senza considerare opportunamente la sua sicurezza; questo avviene a causa delle sempre maggiori pressioni del mercato e/o del management che privilegia il rilascio del software il prima possibile, anche a scapito di feature avanzate (ma spesso essenziali)

Ognuna delle problematiche esposte, se non affrontata attentamente, porta a possibili vulnerabilità ad attacchi di tipo DoS. Ma in concreto, a cosa è necessario prestare attenzione?

PRIMA REGOLA: NON RIVELARE MAI TROPPO

Anche se in crittografia vige la regola per cui non si deve mai fare affidamento sul fatto che il "nemico" (in questo caso chi vuol compromettere il sistema) manchi di informazione (sia essa sul tipo di algoritmi utilizzati sia sul tipo di strumenti) è anche vero che rivelare queste informazioni agevola e, per lo meno, velocizza il compito a possibili malintenzionati. Pertanto è bene "oscurare" qualunque informazione non necessaria al corretto funzionamento dell'applicazione e degli strumenti che ne fanno uso.
Per esempio il server Web potrebbe essere configurato per non rivelare il suo produttore e/o versione; se si usano tool di sviluppo meglio disabilitare tutti i messaggi predefiniti e cambiare nome ai servizi standard o farli rispondere su porte che non sono quelle di default. Un'attenzione particolare va posta nella stampa di messaggi di errore in seguito ad eccezioni: è meglio rivelare il minimo possibile, in modo che un attaccante non possa "estrapolarne" informazioni utili (ovviamente è dannosissimo rivelare password/nomi utente, ma è opportuno filtrare anche messaggi di errore provenienti da dbms o altre componenti in modo da non rivelarne il tipo o la versione e nemmeno il tipo di tabelle in uso o il nome dei server).

ERRORI BLOCCANTI E NON

È necessario prevedere a quali tipi di errori può andare in contro l'applicazione. Ognuno di questi errori va gestito nel modo appropriato. Si supponga, per esempio, che un'applicazione che eroga un certo servizio gestisca un file di log. È inopportuno che errori nella creazione del log vadano ad incidere sulla funzionalità stessa! Allo stesso modo vanno identificati accuratamente tutti i tipi di errore bloccanti e quelli non bloccanti; in seguito è necessario prendere opportune contromisure, sia per segnalarli sia per gestirli. Una particolare cura va posta nella gestione di chiavi progressive persistenti. A livello teorico non esiste mai un numero troppo elevato progressivo prima che un certo programma si blocchi: molto meglio creare dei progressivi "a rotazione" (per esempio progressivi all'interno di un anno o all'interno di un periodo temporale più ristretto).

BUFFER OVERFLOWS E VALIDAZIONE DELL'INPUT

Ogni qualvolta un servizio accetta dei dati dall'esterno deve prevedere un comportamento non destabilizzante in presenza di input errato. In particolare, una delle cause più comuni di vulnerabilità applica-

CGI
(Common Gateway Interface) modalità standard per la generazione di contenuti dinamici sul Web

GARBAGE COLLECTOR
Meccanismo (di solito asincrono e built-in nel linguaggio) per l'effettiva pulizia degli oggetti in memoria non più utilizzabili dall'applicazione.

tiva è la mancata gestione del buffer overflow. Questo avviene ogniqualvolta ci si aspetta una dimensione massima per un determinato parametro o valore e quest'ultimo eccede tale limite. Se non ci sono controlli possono crearsi pericolose falle di sicurezza per tutte quelle applicazioni in cui un simile tipo di errore fa accedere a zone di memoria non protette (non è il caso di Java, ma di solito sono a rischio le applicazioni CGI). Anche la validazione dell'input può presentare problemi. Esistono due tecniche per prevenire un input non valido:

- *validazione positiva*
- *validazione negativa*

la prima identifica tutte e sole le occorrenze valide dell'input (di solito attraverso operazioni di pattern matching o di espressioni regolari), la seconda identifica stringhe errate (per esempio indica se una stringa contiene caratteri non validi). Ai fini della sicurezza è sempre da preferire la prima soluzione, in quanto se identificano tutti gli input validi si è sicuri di non incontrare problemi; è più difficile riuscire a trovare regole generali in modo da definire tutti gli input non validi (di solito si eseguono pattern matching - uno per ogni carattere non valido - ed è facile tralasciarne alcuni). Alcuni problemi legati ad una cattiva gestione della validazione dell'input sono il code injection e il cross side scripting o XSS (per la vastità delle implicazioni connesse e per fornire adeguati esempi e contromisure, questi problemi verranno analizzati in un prossimo articolo).

JMX

JMX sta per Java Management Extensions ed è un'interfaccia standard per il monitoraggio e il controllo delle applicazioni Java

DIFENDERSI DAL CONSUMO DI RISORSE

È già stato evidenziato come è importante adottare le opportune contromisure il prima possibile (si ricordi lo stack OSI). Per questo motivo è possibile impostare opportune regole per limitare l'utilizzo di servizi da parte di una fonte (per esempio si ricorre a regole su firewall o routers). Però, a livello applicativo, è possibile conoscere alcune caratteristiche intrinseche dei servizi offerti che non è possibile conoscere a più basso livello. Un caso esemplificativo si ha quando ci si trova nella situazione in cui un servizio ha la necessità di eseguire numerose computazioni (nonché accessi a risorse esterne quali database, file system di rete o altro) e per questo è particolarmente lento nel fornire i risultati. In questi casi il servizio è potenzialmente a rischio di attacchi DoS ed è necessario prevedere opportune contromisure; il luogo più adatto è a livello applicativo. In questi casi si potrebbe, per esempio:

- limitare il numero di possibili invocazioni al servizio per tutti gli utenti (per esempio non possono essere soddisfatte contemporaneamente più di 10 richieste). Tale gestione può essere fatta con un pool di thread, dove ogni thread risponde ad una richiesta; esauriti i thread a disposizione, eventuali richieste restano in attesa di un thread che si liberi;
- limitare il numero di invocazioni per singolo utente (dove l'utente può essere identificato in maniera "sicura" attraverso l'uso di certificati digitali o in maniera "debole" con uso di username/password o IP address); queste limitazioni sono, di solito, applicate su un certo intervallo di tempo (per esempio non più di 10 invocazioni al minuto).

Soluzioni come queste permettono all'applicazione di non "esaurire" le risorse del sistema e il servizio pur essendo ritardato non va in blocco (si ha un degrado delle prestazioni che non porta a DoS).

Fig. 2: *Riferimenti circolari*

GESTIONE NON CORRETTA DELLE RISORSE

Benché possa sembrare strano, il più delle volte sono le applicazioni stesse ad essere mal progettate e a portare a problemi di esaurimento delle risorse disponibili. Questo porta a un problema DoS anche in mancanza di attacchi specifici, ma in seguito all'uso intensivo (normale) delle applicazioni. Essenzialmente il problema è nel mancato rilascio delle risorse utilizzate o in un cattivo uso dei meccanismi di concorrenza. Nel primo caso si ha un progressivo esaurimento delle risorse; nel secondo caso si possono avere processi in attesa infinita. Il problema più sentito, e tra i più difficili da identificare, è quello relativo al progressivo consumo di memoria dovuto a *memory leak*. Un *memory leak* è una porzione di memoria che risulta in uso anche quando questo non è più vero. Questo avviene per lo più in linguaggi come il C++ che fanno uso di puntatori e che richiedono allo sviluppatore una corretta gestione della memoria, oppure in linguaggi con garbage collector che soffrono di alcune limitazioni (per esempio che usano un contatore per memorizzare il numero di reference agli oggetti: nel caso di riferimenti circolari, esemplificati in **Figura 2**, non riescono a liberare la memoria). Purtroppo anche Java può soffrire di memory leak, nonostante gli algoritmi implementati nel suo garbage collector siano immuni

dal problema delle referenze circolari.

CONTROLLO DELLE RISORSE IN JAVA

Java può essere considerato uno dei principali linguaggi di programmazione lato server per servizi pubblici sul Web. Esistono casi in cui il garbage collector di Java non funziona come dovrebbe e l'applicazione presenta dei memory leak. Infatti il garbage collector riesce a liberare solo quegli oggetti che non sono più referenziati da oggetti attivi. Ma che accade in presenza di variabili con riferimenti attivi ad oggetti con scope (e quindi ciclo di vita) diversi? Un caso concreto: le librerie AWT e i JavaBean fanno uso intensivo del *design pattern Listener.* In pratica se un oggetto *A* è interessato agli eventi di un'istanza *B, A* si registra come ascoltatore *(listener)* di *B*. A questo punto *B* contiene un reference ad *A*. Se *A* ha un ciclo di vita maggiore di *B* e non si elimina il listener (attraverso un opportuno metodo), *B* rimarrà in memoria e non potrà essere recuperato dal garbage collector nemmeno quando il suo ciclo di vita sarà esaurito. Altre fonti di memory leak sono in alcune implementazioni del design pattern *"singleton".* Per una trattazione approfondita si rimanda a **[5]**.
Esistono poi casi in cui è il linguaggio stesso ad avere dei bug che portano a memory leak (è il caso dell'uso di *StringBuffer* bella versione 1.4.1 del linguaggio, *http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4724129).* Un modo per verificare la presenza di memory leak è ricorrere a tool di profilazione come JProbe o altri prodotti commerciali; ma Java ha a disposizione una nuova tecnologia per il controllo delle applicazioni: JMX. La descrizione approfondita della tecnologia esula dagli obiettivi di questo articolo, ma si possono trovare interessanti caratteristiche per il controllo del consumo delle risorse nelle applicazioni che ne fanno uso. Infatti è possibile verificare il consumo della memoria di ogni singola applicazione e di monitorare nel tempo tale consumo; per un esempio concreto si può vedere la libreria di tag JSP *"JMX Memory Monitor",* presente alla pagina *http://dev2dev.bea.com/codelibrary/code/jmx_memory.jsp*).

IMPORTANZA DEL TEST E DELL'AGGIORNAMENTO

I test non debbono essere solo funzionali: è necessario preventivare possibili accessi non autorizzati, consumi di risorse di almeno un fattore oltre a quello massimo previsto e input diversi da quelli d'uso normale. Purtroppo non è possibile, ad oggi, prevedere dei test esaustivi per coprire tutte le possibili falle di sicurezza ma è sempre consigliabile, a livello aziendale, predisporre dei test automatici standard per le più comuni casistiche e, in particolare, test di carico e test di modulo con diversi input. Tra le regole d'oro della sicurezza c'è poi quella di mantenere aggiornati tutti i componenti software del sistema (dal sistema operativo, all'application server fino alla piattaforma software).

MINIMO PROFILO

Infine è da segnalare un consiglio sempre valido: mai permettere all'applicazione di operare su porzioni di sistema non necessarie e mai permetterle accesso a risorse non preventivate. È sempre possibile che un'applicazione o una sua componente contengano dei bug che possono portare ad una vulnerabilità. È buona norma accertarsi che qualunque applicazione sia eseguita con privilegi minimali e strettamente necessari al suo corretto utilizzo.
Questo ovviamente per tutti gli ambienti enterprise dove sia gli utenti che le applicazioni possiedono propri profili. È assolutamente sconsigliabile che una qualsiasi applicazione che colloqui con il mondo esterno abbia diritti di super user o root.

CONCLUSIONI

Come si è visto nell'introduzione dell'articolo attacchi di tipo DoS sono solo un esempio dei possibili attacchi ad un server Web. Questa notevole diversità di tipi di attacchi porta a definire, nell'insieme, un certo numero di contromisure, ognuna più appropriata per contrastare determinate azioni ma, nel complesso, coordinate e il più possibile complete. In letteratura è comune trovare il termine *"defence in depth"* (**Figura 3**) per indicare un piano strategico di possibili contromisure, ognuna focalizzata ad un diverso livello di astrazione e di implementazione. In figura si possono notare le diverse contromisure comunemente adottate ai vari livelli **[3]**.

Ivan Venuti

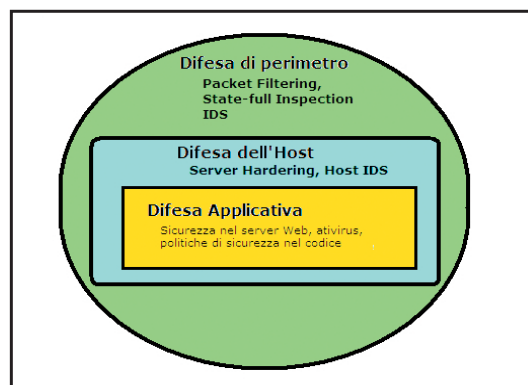



Fig. 3: defence in depth

[1] A SUMMARY OF DOS/DDOS PREVENTION, MONITORING AND MITIGATION TECHNIQUES IN A SERVICE PROVIDER ENVIRONMENT
M. Glenn
(SANS Institute)
2003

[2] WHAT DO WE MEAN BY NETWORK DENIAL OF SERVICE?
C. Shields
http://www.cs.georgetown.edu/~clay/research/pubs/shields-ndos.pdf

[3] WEB SERVER SECURITY GUIDELINES
(CERT-In)
2004
CISG-2004-04.pdf

[4] APPLICATION DEVELOPMENT TECHNOLOGY AND TOOLS: VULNERABILITIES AND THREAT MANAGEMENT WITH SECURE PROGRAMMING PRACTICES, A DEFENSE IN-DEPTH APPROACH
V. L Ankolekar
GSEC
2003

[5] PLUGGING MEMORY LEAKS
T.K.T. Leung
(JavaPro)
giugno 1999

Un sistema GPS con mappa geografica

Mettiamo la mappa nel GPS

Avendo a disposizione un GPS, ci divertiremo a realizzare un sistema di acquisizione dei dati e a disegnare la traccia degli spostamenti su una mappa

Utilizza questo spazio per le tue annotazioni

Siamo giunti al terzo appuntamento dedicato alla realizzazione di un sistema completo di interfacciamento con ricevitori satellitari GPS: nel primo articolo abbiamo analizzato le principali funzioni del protocollo NMEA, utilizzato per lo scambio di informazioni tra apparati di navigazione, abbiamo descritto inoltre la realizzazione di un sistema completo anche dal punto di vista elettronico, utilizzando il modulo ricevitore satellitare GEOGAP. Nel secondo appuntamento è stato realizzato un sistema di acquisizione dei dati di posizione e quota e degli spostamenti di una piattaforma dotata di GPS. In questo terzo incontro proporremo le metodologie di inserimento e gestione di una cartina geografica, che permetta la visualizzazione grafica della posizione, oltre che dei dati numerici di latitudine, longitudine geografica e di quota.

LE CARTE GEOGRAFICHE

Dal punto di vista geometrico possiamo pensare ad una carta geografica, come ad una proiezione della rappresentazione sferica del nostro pianeta rispetto ad una superficie piana. Supponiamo di avere a disposizione un mappamondo dotato di illuminazione interna e di avvicinare ad esso una serie di fogli di carta piegati in modo appropriato. La prima idea che ci verrebbe sarebbe probabilmente quella di avvicinare al nostro ipotetico proiettore terrestre un foglio di carta disposto in senso piano, ma ben presto scopriremmo che la rappresentazione della proiezione della sfera sulla carta geografica, sarebbe accurata in un solo punto: questa rappresentazione viene infatti utilizzata per le carte polari, dove i meridiani convergono nel polo geografico e le linee di uguale latitudine, sono quindi rappresentate come cerchi, questo particolare tipo di mappa si chiama a proiezione stereografica polare. Supponiamo di piegare il nostro foglio fino a realizzare un cilindro di diametro uguale alla sfera dell'ipotetico proiettore terrestre e di avvolgervi il mappamondo luminoso. La proiezione che otterremmo una volta disegnati i contorni sul foglio, e dopo averlo srotolato, avrebbe meridiani e paralleli perpendicolari tra loro come su di un foglio di carta millimetrata. La proiezione in questione si dice di Mercatore, che ha svariate caratteristiche molto interessanti, tra le quali quella di potere effettuare una conversione diretta tra coordinate geografiche e posizione cartesiana sulla mappa: nella nostra applicazione utilizzeremo questo tipo di carte geografiche. Dal punto di vista pratico è possibile scaricare molte mappe dalla rete, ovviamente registrandosi ed acquistando le carte, che nella maggior parte dei casi sono protette da diritti di copyright.

IL NOSTRO PROGRAMMA

Siul sito di ioProgrammo all'interno della sezione doznload, utilizzando le password contenute nella pagina dell'editoriale, potrete scaricare il programma completo a cui facciamo riferimento in questo articolo, in cui sono elaborate anche le funzioni di acquisizione dati dal modulo GPS che abbiamo trattato in precedenza. Al momento della ricezione dei dati dal GPS, lanciando il programma sul PC ed effettuando la connessione al GPS, tramite l'opzione 'Connect to GPS' dal menu GPS, si nota il flusso di informazioni NMEA nella finestra 'NMEA Raw informations', in questo articolo spiegheremo comle visualizzare le coordinate e la posizione su una mappa come a proiezione cilindrica di Mercatore: questo tipo di mappe sono scaricabili da molti siti web, previa registrazione e talvolta a seguito del pagamento dei rispettivi diritti di copyright.

È buona norma utilizzare mappe che diano delle indicazioni precise sui limiti degli estremi delle coordinate della mappa, questo ci servirà infatti per calibrare correttamente i riferimenti.

IL CODICE SORGENTE

Come di consueto cercheremo di produrre un codice semplice e lineare, preferendo senz'altro in questa sede la chiarezza di esposizione eventualmente anche rispetto all'ottimizzazione del programma, per ovvi scopi didattici. Per l'acquisizione dei dati viene usato l'ottimo componente freeware '*CPDrv .pas*' che può essere scaricato gratuitamente dal Web *(Cdd4.zip)*. La procedura *FormCreate*, eseguita al momento della creazione della form principale, ha lo scopo di visualizzare la finestra iniziale contenente licenza e disclaimer, di inizializzare le variabili di posizione del sistema ed infine predisporre la forma grafica della form. Il codice proposto é il seguente:

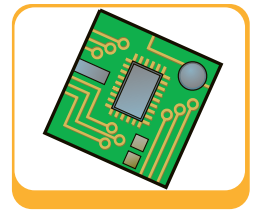
```
procedure TGPSPositionPlotterMainForm.FormCreate(
                                        Sender: TObject);
var
GPSPositionPlotterAboutdlg: TGPSPositionPlotterAboutBox;
I,Xc,Yc:Integer;
begin
  // About Dialog
  GPSPositionPlotterAboutdlg := nil;
  GPSPositionPlotterAboutdlg :=
TGPSPositionPlotterAboutBox.Create( Self );
  GPSPositionPlotterAboutdlg.ShowModal;
  // Initial Position
  InitialPositionLat:=45;  InitialPositionLon:=15;
  Startup:=True;
  //Graphic Map
  CurrentFile:='Mondo.BMP';
  Mapscale:=1000;   //1 pixel equal to 1,8 mt
  MapImage.Picture.LoadFromFile(CurrentFile);
  MapImage.Canvas.Pen.Color:=clYellow;
  MapImage.Canvas.Brush.Style:=bsClear ;
  xc:=( MAPPanel.Width div 2);
  yc:=(MAPPanel.Height div 2);
  For I:=0 to 8 do begin
    MapImage.Canvas.Ellipse((Xc-(I*50)),(
                    Yc-(I*50)),(Xc+(I*50)),(Yc+(I*50)));
  end;
  //Map Coordinates
  MapExtremitiesChange;
end;
```

Notate l'istruzione *"CurrentFile:= 'Mondo.BMP';"* che prelude all'utilizzo di carte geografiche in formato bitmap e che predispone il sistema ad aprire la mappa di default rappresentante tutto il globo terrestre attraverso l'istruzione *"MapImage.Picture .LoadFromFile(CurrentFile);"*.

Qualora l'utente decidesse di ridimensionare la finestra, viene eseguito l'event handler riportato di seguito che provvede semplicemente a ricaricare l'immagine relativa alla mappa corrente ed a ridisegnare la sequenza di cerchi concentrici gialli di riferimento.

```
procedure TGPSPositionPlotterMainForm.FormResize(
                                        Sender: TObject);
Var
I,Xc,Yc:Integer;
begin
  MapImage.Picture.LoadFromFile(CurrentFile);
  MapImage.Canvas.Pen.Color:=clYellow;
  MapImage.Canvas.Brush.Style:=bsClear ;
  xc:=( MAPPanel.Width div 2);
  yc:=(MAPPanel.Height div 2);
  For I:=0 to 8 do begin
    MapImage.Canvas.Ellipse((Xc-(I*50)),(
                    Yc-(I*50)),(Xc+(I*50)),(Yc+(I*50)));
  end;
end;
```

Qualora l'utente cambi i valori geografici di riferimento della mappa, viene eseguita una delle quattro procedure che seguono, corrispondenti agli event handler dell'evento "*OnChange*" di ciascuna delle quattro caselle di edit poste sul lato sinistro della form principale, corrispondenti alle coordinate limite della cartina. Viene eseguita in ogni caso la procedura "*MapExtremitiesChange*" descritta poco più avanti. La procedura *"MapExtremitiesChange"* provvede a convertire, innanzi tutto, una stringa contenuta in quattro caselle d'edit contenenti i valori delle coordinate, delle estremità della cartina, in valori numerici: se la conversione non è andata a buon fine viene visualizzato un messaggio d'errore.
Il valore delle coordinate delle estremità della mappa vengono memorizzate nelle variabili globali *LonLeft* (Longitudine al lato sinistro), *LonRight* (Longitudine al lato destro), *LatTop* (Latitudine al margine superiore) e *LatBottom* (Latitudine al margine inferiore).

```
procedure TGPSPositionPlotterMainForm.
                                  MapExtremitiesChange;
Var Code:Integer;
begin
  //Manages Changes upon Map Parameters
  Val(MapLatTopEdit.Text, lattop, Code);
  //Error during conversion to integer?
  if Code <> 0 then
    MessageDlg('Map coordinate Error:
              ' + IntToStr(Code), mtWarning, [mbOk], 0);
  Val(MapLatBottomEdit.Text, latbottom, Code);
  //Error during conversion to integer?
  if Code <> 0 then
    MessageDlg('Map coordinate Error:
```

COMPONENTI ELETTRONICI

REALIZZAZIONE CON MODULO GEOGAP

N1	PCExplorer light
N1	Modulo Elisys GEOGAP
N2	Transistor BC 107
N2	Resistenza 1Kohm 1/4W
N2	Resistenza 10Kohm 1/4W
N2	Resistenza 33Kohm 1/4W
N1	Condensatore 100 nF

REALIZZAZIONE CON GPS

N1	PCExplorer light
N1	GPS commerciale
N1	Cavo connessione (dipende dal modello di GPS)

NOTA

ACQUISTARE PC EXPLORER LIGHT

L'apparecchiatura PC Explorer light è prodotta e commercializzata dalla Elisys s.r.l. e può essere acquistata al prezzo di € 213,60 nella versione light, € 99 nella versione basic e € 69 in kit (IVA inclusa) sul web all'indirizzo www.pcexplorer.it oppure inviando una e-mail a: pcexplorer@elisys.it, od anche telefonicamente al numero 0823/468565 o via Fax al: 0823/495483.

```
                 ' + IntToStr(Code), mtWarning, [mbOk], 0);
  Val(MapLonLeftEdit.Text,lonleft, Code);
  //Error during conversion to integer?
  if Code <> 0 then
    MessageDlg('Map coordinate Error:
                 ' + IntToStr(Code), mtWarning, [mbOk], 0);
  Val(MapLonRightEdit.Text,lonright, Code);
  //Error during conversion to integer?
  if Code <> 0 then
    MessageDlg('Map coordinate Error:
                 ' + IntToStr(Code), mtWarning, [mbOk], 0);
end;
```

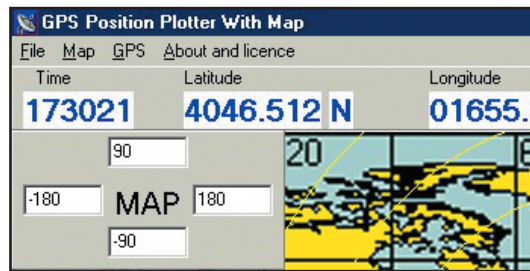



Fig. 1: *I dati della cartina possono essere variati per mezzo di quattro caselle di dialogo*

La procedura di gestione e lettura delle coordinate geografiche da parte del GPS, è stata adattata alle nuove funzionalità relative all'utilizzo delle carte geografiche. In sostanza questo frammento di codice è stato variato nel calcolo della posizione da disegnare sulla mappa attraverso le istruzioni seguenti:

```
// ***NEW*** Position Plot
X:=Trunc(MapImage.Picture.Width/(LonRight-LonLeft)
              *((LonGradi+(LonPrimi/60)-LonLeft)));
Y:=MapImage.Picture.Height-Trunc(
         MapImage.Picture.Height/(LatTop-LatBottom)
              *(LatGradi+(LatPrimi/60)-LatBottom));
XYLabel.Caption:='Map X:'+Inttostr(x)+'   Map Y:'+
                                         inttoStr(y);
```

È possibile notare che in questa fase si tiene conto delle coordinate relative ai margini dell'immagine, individuate dalle variabili *LonLeft* (Longitudine al lato sinistro), *LonRight* (Longitudine al lato destro), *LatTop* (Latitudine al margine superiore) e *LatBottom* (Latitudine al margine inferiore). La posizione rilevata dal GPS (Latitudine e Longitudine) è stata suddivisa in gradi e primi, utilizzando le variabili *LatGradi, LatPrimi, LonGradi, LonPrimi* dal significato abbastanza ovvio. I valori di *X* e *Y* riferiti alla mappa vengono poi successivamente utilizzati per disegnare un cerchio rosso pieno della dimensione di 7 pixel di raggio centrato sulla rappresentazione grafica della posizione individuata sulla mappa.
Per completezza e comodità di lettura si riporta la procedura nella sua totalità qui di seguito.

```
procedure
TGPSPositionPlotterMainForm.NMEApresentPosition;
 var
  stringline,s:string[255];
  io,fo,ila,fla,ilo,flo,ialt,falt,v,m,code,X,Y:Integer;
  ora,lat,ns,lon,ew,quota,fm:string;
  feasible:Boolean;
  gra,pri:real;
LatReal,LonReal,Quotareal,LatGradi,LatPrimi,LonGradi,
                                          LonPrimi:Real;
begin
  Stringline:=GlobalString;
  begin
    (*Decodifico il formato NMEA GPGGA Global Position*)
    If copy(Stringline,1,6)='$GPGGA' then begin
      io:=0; fo:=0; ila:=0; fla:=0; ilo:=0; flo:=0; ialt:=0;
      falt:=0; v:=0; ora:=''; lat:=''; lon:=''; ns:='';
                                              quota:='';
      fm:='';
      Feasible:=True;
      for m:=1 to (length(stringline)-7) do begin
        If copy(Stringline,m,1)=',' then v:=v+1;
        if v=0 then io:=m; if v=1 then fo:=m;
                                   if v=1 then ila:=m;
        if v=2 then fla:=m; if v=3 then ilo:=m;
                                   if v=4 then flo:=m;
        if v=8 then ialt:=m; if v=9 then falt:=m;
        (* Controllo se il GPS sta tracciando oppure no*)
        if fo>(io+1) then begin
          ora:=copy(Stringline,(io+2),(fo-io-1));
          Feasible:=True;
        end
        else
        begin
          Feasible:=False;
        end;
        if fla>(ila+1) then begin
          lat:=copy(Stringline,(ila+2),(fla-ila-1));
          Val( Lat,LatReal,code);
          ns:=copy(Stringline,(fla+2),1);
          Feasible:=True;
        end
        else
        begin
```

NOTA

CONNESSIONE SERIALE:

Il programma funziona acquisendo i dati dal GPS dalla porta seriale COM1, 9600 baud, 8 bit dati, 1 bit di Stop, se sul proprio sistema si intende utilizzare una porta differente, oppure parametri diversi da quelli citati è sufficiente variare i parametri di connessione del componente 'SerialDriver' attraverso l'Object Inspetor di Delphi.

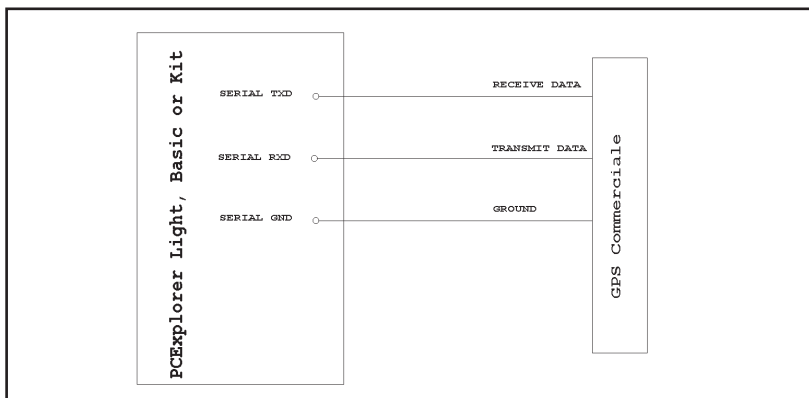



Fig. 2: *Lo schema elettrico comprende la semplice connessione GPS-PCExplorer light tramite il cavo in dotazione al GPS*

```
    Feasible:=False;
  end;
  if flo>(ilo+1) then begin
    lon:=copy(Stringline,(ilo+2),(flo-ilo-1));
    Val( Lon,LonReal,code);
    ew:=copy(Stringline,(flo+2),1);
    Feasible:=True;
  end
  else
  begin
    Feasible:=False;
  end;
  if falt>(ialt+1) then begin
    quota:=copy(Stringline,(ialt+2),(falt-ialt-1));
    fm:=copy(Stringline,(falt+2),1);
    Feasible:=True;
  end
  else
  begin
    Feasible:=False;
  end;
end;
if feasible then begin
   inc(GlobalRetrivedNMEAString);
   s:='NMEA Global Position and Altitude
        at:'+ora+' '+lat+' '+ns+' '+lon+' '+ew+'
                                  '+quota+' '+fm;
   label3.caption:=ora; label4.caption:=lat;
                              label9.caption:=ns;
   label8.caption:=lon; label5.caption:=ew;
                              label6.caption:=quota;
   label7.caption:=fm;
   // Graphic plot
   if (Startup and Feasible) then begin
                                //First Coordinate
        InitialPositionLat:=LatReal; //Map Centre
        InitialPositionLon:=LonReal;
        Startup:=False;
   end;
   LatGradi:=Trunc(LatReal/100);
   LonGradi:=Trunc(LonReal/100);
   LonPrimi:=(LonReal-(LonGradi*100));
   LatPrimi:=(LatReal-(LatGradi*100));
   // OLD Position Plot
   //Y:= ( MAPPanel.Height div 2 )+Trunc((
         InitialPositionLat-LatReal)*MapScale);
   //X:= ( MAPPanel.Width div 2 )+Trunc((
         LonReal-InitialPositionLon)*MapScale);
   // ***NEW*** Position Plot
  X:=Trunc(MapImage.Picture.Width/(
  LonRight-LonLeft)*((LonGradi+(LonPrimi/60)-
                                        LonLeft)));
   Y:=MapImage.Picture.Height-Trunc(
   MapImage.Picture.Height/(LatTop-LatBottom)
        *(LatGradi+(LatPrimi/60)-LatBottom));
   XYLabel.Caption:='Map X:'+Inttostr(x)+'
                               Map Y:'+ inttoStr(y);
   MapImage.Canvas.Pen.Color:=clRed;
```

```
          MapImage.Canvas.Brush.Color:=clRed;
          MapImage.Canvas.Brush.Style:=bsSolid;
          MapImage.Canvas.Ellipse((x-7),(y-7),
                                        (x+7),(y+7));
        end;
      end;
    end;
  end;
```

Una nota finale è meritata dalle procedure di connessione: il programma funziona utilizzando la porta seriale COM1, che in alcuni PC viene utilizzata per la gestione del mouse, in questo caso per evitare conflitti di sistema è necessario variare i parametri di connessione del componente *'SerialDriver'*. I parametri di comunicazione della porta seriale vengono impostati come: COM1, 9600 baud, 8 bit dati, 1 bit di Stop: se sul proprio sistema si intende utilizzare una porta differente, oppure parametri diversi da quelli citati è sufficiente variare i parametri di connessione del componente *'SerialDriver'* attraverso l'*Object Inspetor* di Delphi.

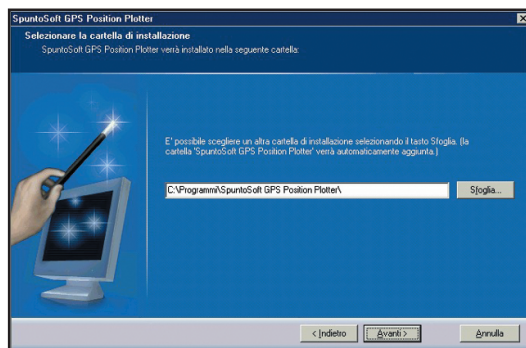
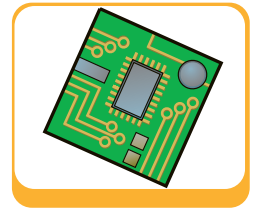



Fig. 12: *L'installazione del software di gestione avviene semplicemente eseguendo l'apposito programma*

CONCLUSIONI

Il semplice utilizzo di un ricevitore satellitare integrato GPS, oppure di un qualunque modello commerciale dotato di una porta d'uscita seriale, unitamente al software presentato in questa sede e negli appuntamenti precedenti sono sufficienti per realizzare un sistema di misurazione della posizione, dotato di mappe geografiche. Le applicazioni sono innumerevoli, dai sistemi per auto e per barca ai quali siamo ormai abituati, ma anche nel campo dell'ingegneria civile e della topografia. Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

Il sistema proposto in queste pagine è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light':

ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo:

'http://www.pcexplorer.it'
oppure
'http://web.tiscali.it/spuntosoft/'
od infine inviare una e-mail a:
spuntosoft@tiscali.it.

L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo:
luca.spuntoni@ioprogrammo.it

Gestire i dati sul Web con ASP.NET

Data Controls e l'accesso ai dati

L'accesso ai dati è una delle operazioni più diffuse nelle applicazioni web ed ASP.NET offre un comodo sistema, che si basa su ADO.NET tale a rendere facilmente implementabili soluzioni anche molto complesse!

L'accesso ai dati è una componente fondamentale di qualsiasi applicazione web, dato che nel 90% dei casi ciò che uno sviluppatore fa non è altro che leggere o scrivere. Nell'arco degli ultimi anni, infatti, è diventato sempre più frequente manipolare informazioni salvate in formato XML, oltre che quelle classiche in un database. ASP.NET, tecnologia moderna e destinata ad applicazioni enterprise, non avrebbe potuto ignorare l'evoluzione dell'accesso ai dati in questo senso. Per questo quando si parla di accesso ai dati con ASP .NET, ci riferiamo ad una qualsiasi fonte dati, sia esso un database oppure anche un più esotico file XML.

ALLA BASE C'È ADO.NET

Acronimo di *ActiveX Data Objects*, ADO è un insieme di librerie pensate per favorire l'accesso ai dati. Quello che unisce ADO e ADO.NET è lo scopo: unificare le metodologie di accesso ai dati a prescindere dalla fonte e dal linguaggio utilizzato. Ciò che divide queste due tecnologie è tutto il resto, in particolar modo la metodologia d'accesso ai dati.

L'APPROCCIO DISCONNESSO: IL DATASET

Le differenze principali tra l'approccio di ADO rispetto a quello offerto da ADO.NET non si limitano alla comparsa di classi create appositamente per ogni database, in modo da migliorarne le performance, ma soprattutto modificano la modalità di accesso ai dati in senso "disconnesso". Questo approccio privilegia dunque una "fotografia" dei dati, anziché un accesso sequenziale tipico del recordset di ADO. In realtà in ADO.NET sono previste entrambe le modalità di accesso, ma di sicuro la più utilizzata è quella che prevede l'utilizzo di un DataSet.

http://localhost/ioprogrammo/03/dg2.aspx - Microsoft Internet Explorer

title	link	pubDate	description	creator	category
ASP to ASP.NET Migration Roadshow in 11 città	http://www.aspitalia.com/servizi/focus/mostra.aspx?ID=586	Thu, 03 Mar 2005 00:00:00 +0100	Un altro appuntamento gratuito da non perdere. Microsoft Italia organizzerà un roadshow dedicato alla migrazione di applicazioni ASP verso ASP.NET articolato in 12 date differenti, che toccheranno...	Daniele Bochicchio	FocusON
#674 - Effettuare il postback su un'altra pagina con ASP.NET 2.0	http://www.aspitalia.com/liste/usag/script.aspx?ID=674	Fri, 04 Feb 2005 00:00:00 +0100	Una delle novità introdotte da ASP.NET 2.0 è la possibilità di effettuare il post di una pagina verso un'altra pagina. Successivamente, dalla pagina chiamata si può accedere, tramite l'oggetto ...	Stefano Mostarda	unoscript@l
#673 - Client CallBack con ASP.NET 2.0	http://www.aspitalia.com/liste/usag/script.aspx?ID=673	Thu, 03 Feb 2005 00:00:00 +0100	Con l'avvento di ASP.NET 2.0 verranno introdotte molte nuove funzionalità, tra le quali il Client Callback, ossia la possibilità di richiamare un metodo lato server attraverso un codice Javascript (...	Massimo Cappellesso	unoscript@l
			Dal 31 Gennaio al 04 Febbraio. Fra il 5		

Operazione completata

Fig. 1: **Repeater mostra i dati contenuti in un template**

Come il nome stesso suggerisce, la classe *DataSet* è un contenitore di informazioni neutro, rispetto alla fonte dati, di informazioni. È composto da n *DataTable*, ovvero da tabelle, che a loro volta sono composti da *DataColumns* (colonne) e *DataRows* (righe). In genere una *DataTable*, è riempita attraverso un *DataAdapter*, che non è altro che una classe particolare che funge da ponte tra i dati veri e propri ed il DataSet. La particolarità del *DataAdapter* è che racchiude in sé tutta la logica necessaria ad aprire la connessione al database, effettuare la query, inserire i dati all'interno del *DataTable* e chiudere la connessione. Il vantaggio del DataSet rispetto alla lettura sequenziale, che resta supportata, è che diventa possibile "spostare" i dati che sono contenuti all'interno della collection più facilmente. Di fatto la collection è già popolata con tutti i dati che servono e non è più necessario ritornare alla fonte per recuperarli. D'altro canto in questo modo appare chiaro che i dati, una volta che il DataSet è stato popolato, possono essere differenti da quelli contenuti nella fonte dati anche solo dopo pochi attimi.

COME IL VECCHIO RECORDSET: IL DATAREADER

In realtà il *DataAdapter*, utilizza un metodo chiamato *Fill* per ottenere le informazioni necessarie a riempire il *DataSet*. *Fill* non fa altro che istanziare una serie di oggetti per arrivare al proprio scopo. Tra questi spicca il *DataReader,* che in alcuni casi può essere referenziato anche come *SqlDataReader* o *OleDbDataReader* a seconda del data provider utilizzato. Si tratta di una classe che permette un accesso in sola lettura, in modalità *forward only* (cioè, solo in avanti) ai dati, un po' come il recordset di ADO. Rispetto a quest'ultimo però ha diverse funzionalità in meno, ad esempio, non permette di utilizzare cursori particolari né di modificare i dati. Per posizionarsi sulla riga che si vuole leggere è sufficiente utilizzarne il metodo *Read,* mentre per le altre operazioni l'accesso alle informazioni è del tutto simile alle tecniche già utilizzate per il *Recordset* di ADO.

MOSTRARE I DATI: IL DATABINDING

Dopo questa rapida occhiata alle classi che sono alla base di ADO.NET, è giunta l'ora di mostrare come ASP.NET sfrutta queste classi per mostrare i dati all'interno di una pagina. Alla base di tutto c'è un meccanismo, supportato da molti dei controls di ASP.NET, il *Databinding.* Letteralmente vuol dire "associazione di dati" ed infatti è un sistema con il quale si associano i dati estratti da una fonte dati ad un contenitore disposto in qualche punto sulla pagina. Questa fonte può essere un *DataSet*, una *DataTable*, un *DataReader,* ma più in generale una qualsiasi classe che implementi una tra le interfacce IList, *IListSource* e *IEnumerable.* È così possibile fare il binding anche di collection personalizzate. L'associazione è effettuata utilizzando una classe particolare, chiamata *DataItem*, che fa da ponte verso il contenitore verso e proprio dei dati:

```
<%#Container.DataItem%>
```

L'insieme dei caratteri *<# %>* dice al compilatore che si tratta di un'istruzione da utilizzare in fase di databinding.

I DATACONTROLS

Senza controls da utilizzare per visualizzare i dati, il databinding sarebbe inutile. Senza entrare troppo nei dettagli, perché esula dallo scopo di questo articolo, i *Data Controls* sono un insieme di controlli che permettono di associare i dati, sfruttando il meccanismo del *Databinding,* e visualizzarli nella

title_id	title	type	pub_id	price	advance	royalty	ytd_sales	notes	pubdate
BU1032	The Busy Executive's Database Guide	business	1389	19,9900	5000,0000	10	4095	An overview of available database systems with emphasis on common business applications. Illustrated.	12/06/1991 0.00.00
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	1389	11,9500	5000,0000	10	3876	Helpful hints on how to use your electronic resources to the best advantage.	09/06/1991 0.00.00
BU2075	You Can Combat Computer Stress!	business	0736	2,9900	10125,0000	24	18722	The latest medical and psychological techniques for living with the electronic office. Easy-to-understand explanations.	30/06/1991 0.00.00
BU7832	Straight Talk About Computers	business	1389	19,9900	5000,0000	10	4095	Annotated analysis of what computers can do for you: a no-hype guide for the critical user.	22/06/1991 0.00.00
MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19,9900	0	12	2032	Favorite recipes for quick, easy, and elegant meals.	09/06/1991 0.00.00
MC3021	The Gourmet Microwave	mod_cook	0877	2,9900	15000,0000	24	22246	Traditional French gourmet recipes adapted for modern microwave cooking.	18/06/1991 0.00.00
PC1035	But Is It User Friendly?	popular_comp	1389	22,9500	7000,0000	16	8780	A survey of software for the naive user, focusing on the 'friendliness' of each.	30/06/1991 0.00.00
PC8888	Secrets of Silicon Valley	popular_comp	1389	20,0000	8000,0000	10	4095	Muckraking reporting on the world's largest computer hardware and software manufacturers.	12/06/1994 0.00.00
PC9999	Net Etiquette	popular_comp	1389					A must-read for computer conferencing.	06/08/2000 1.33.54
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	0877	21,5900	7000,0000	10	375	A must for the specialist, this book examines the difference between those who hate and fear computers and those who don't.	21/10/1991 0.00.00
PS2091	Is Anger the Enemy?	psychology	0736	10,9500	2275,0000	12	2045	Carefully researched study of the effects of strong emotions on the body. Metabolic charts included.	15/06/1991 0.00.00
PS2106	Life Without Fear	psychology	0736	7,0000	6000,0000	10	111	New exercise, meditation, and nutritional techniques that can reduce the shock of daily interactions. Popular audience. Sample menus included, exercise video available separately.	05/10/1991 0.00.00

Fig. 2: ***Un datagrid consente possibilità di ordinamento e paginazione***

pagina, sfruttando dei template per ripetere gli stessi. I controls principali di questa famiglia sono tre:

- **Repeater:** è il più semplice di tutti e fa esattamente una sola cosa molto semplice, ovvero ripetere i dati contenuti nel template;
- **DataList:** è una via di mezzo tra i controls presi in esame ed offre funzionalità più avanzate rispetto al repeater, lasciando comunque una certa libertà di personalizzazione;
- **DataGrid:** è il più completo tra quelli presenti, offre funzionalità di ordinamento, paginazione e modifica già incluse e facilmente implementabili.

La scelta tra uno di questi è dettata essenzialmente delle caratteristiche offerte, perché ad esempio il DataGrid ha meccanismi già pronti per paginazione ed ordinamento dei dati, mentre gli altri ne sono privi e per dotarli di queste funzionalità è necessario

L'AUTORE

Daniele Bochicchio è il content manager di *ASPItalia.com*, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP .NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

COM'È FATTO UN MANAGED PROVIDER

Un managed provider è composto da un insieme di classi, raggruppate per ordine e semplicità all'interno di un namespace particolare. Sono disponibili delle interfacce all'interno del namespace *System.Data .Comm* che sono utilizzate da tutti gli altri per costruire le implementazioni concrete proprio di ogni provider:

IDbConnection
IDbCommand
IDbDataReader
IDbDataAdapter

Il punto di partenza di ciascun data provider è dunque creare quattro oggetti che implementino queste interfacce. In genere questi oggetti sono chiamati *Connection, Command, DataReader* e *DataAdapter* e sono preceduti da un prefisso specifico per il data provider.
Ad esempio le classi proprie di SQL Server avranno il prefisso Sql nel nome e si trovano nel namespace *System.Data.SqlClient,* così come quelle OLEdb si riconoscono dal prefisso OleDb e sono conteneute nel namespace *System .Data.OleDb.*

scrivere molto più codice aggiuntivo. Per contro il DataGrid produce una tabella HTML, mentre il DataList e Repeater consentono flessibilità e spesso sono utilizzati (specie quest'ultimo) per produrre anche output di solo testo o in formato XML.
Ciò che hanno in comune è la proprietà *DataSource,* che serve per specificare la sorgente dei dati, un metodo *DataBind()* per associare questi dati, e tre eventi:

- **ItemCreated:** scatenato all'aggiunta di ogni singolo item al *Data Control;*
- **ItemDataBound:** scatenato quando viene effettuata l'associazione dei dati su ogni singolo item;
- **ItemCommand:** scatenato quando un control contenuto in un item richiede l'esecuzione di un comando, in genere associato alle funzionalità di modifica o cancellazione.

Fig. 3: ***Una tabella può essere formattata usando dei template***

I *controls* condividono anche l'associazione dei dati, poiché ognuno di questi è infatti a suo volta composto da un numero di controls identificati dal suffisso Items presente nel nome. Per il DataGrid abbiamo la collezione *DataGridItems,* per il *DataList DataListItems* e per il Repeater *RepeaterItems.* Quando associamo al control la fonte dati, usando la proprietà DataSource, ed effettiamo il binding con il metodo *DataBind(),* viene enumerata la collection con i dati, attraverso un ciclo su ogni elemento che quest'ultima contiene. Nel caso di un DataSet, viene fatto un ciclo sulla collezione *Rows* contenuta nel DataTable di default, al cui interno ha una serie di instanze di *DataRow,* che altro non sono che riferimenti ad un record estratto dal database. Quindi, per ogni *DataRow* è creato un RepeaterItem (nel caso di un *Repeater*) o un *DataGridItem* (nel caso di un *DataGrid*) e viene invocato l'evento *ItemCreated,* associando la *DataRow* alla proprietà *DataItem* del *RepeaterItem.* Viene quindi invocato l'evento *ItemDataBound* ed il ciclo continua con le righe successive, fino all'esaurimento dei dati contenuti nella collection.

UN DATAGRID IN 2 PASSI

Editate un file con il notepad o con un qualunque editor di testo, riempitelo al suo interno con il contenuto estratto dai passi seguenti e salvatelo nella root di IIS con il nome rss.aspx. Provate poi ad eseguire l'applicazione puntando il browser su *http://localhost/rss. aspx.* Quello che facciamo in questo esempio è collegare un controllo di tipo Repeater a un file XML contenente delle news in formato RSS.
Tramite il control le mostriamo a video in modo ordinato, sfruttando le capacità di lettura che quest'ultimo offre da qualsiasi fonte dati.

LETTURA DI XML CON IL DATASET

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<SCRIPT  language="C#" runat="server">
void Page_Load (Object sender, EventArgs e) {
   // creo il dataset
   DataSet ds= new DataSet();
   // leggo i dati dal file XML
   ds.ReadXml(Server.MapPath("feed.xml"));
   // estraggo tutti i nodi di nome item
   rp.DataSource = new DataView(ds.Tables["item"]);
   rp.DataBind();
}
</SCRIPT>
```

1 **Creiamo un nuovo oggetto di tipo *Dataset* chiamato ds. L'oggetto *ds* viene riempito con il contenuto del file *feed.xml* grazie al metodo *ReadXml*. Viene poi creato un *Repeater* a cui andremo ad associare tutti i dati estratti dal file XML, salvato nell'oggetto ds ed infine viene richiamato il metodo *DataBind* che fisicamente effettua il collegamento dei dati al controllo *rp* e li mostra a video.**

ASSOCIAMO IL DATASET AL DATA CONTROL

```
<ASP:repeater id="rp" runat="server">
<ItemTemplate>
<div style="border:1px dotted gray; background-color: #f0f0f0; margin: 5px;">
<p><strong><a href="<%#DataBinder.Eval(Container.DataItem, "link")
 %>"><%#DataBinder.Eval(Container.DataItem, "title")%></a></strong><br />
<%#DataBinder.Eval(Container.DataItem, "pubDate")%> -
                              <%#DataBinder.Eval(Container.DataItem, "creator")%>
</p>
<p><%#DataBinder.Eval(Container.DataItem, "description")%></p>
</div>
</ItemTemplate>
</ASP:repeater>
```

2 **Viene definito un controllo di titpo *Repeater* associato all'oggetto *rp*. Questo controllo mostrerà il contenuto dei campi *"link"*, *"pubDate"*, *"title"* e *"creator"*.**

COME PERSONALIZZARE L'OUTPUT: I TEMPLATE

Per personalizzare l'output associato ad un control, come già detto, si sfruttano i template. In realtà c'è un Data Control, tra tutti, in grado di estrarre i dati

senza specificare nessun template, ed è il DataGrid. Per tutti gli altri, invece, è necessario specificare almeno un template ed in modo specifico almeno l'*ItemTemplate*. Un template non è altro che un control che al proprio interno può contenerne di altri ed i tipi supportati dai *Data Controls*, il cui nome ne ricorda le funzionalità, sono:

- AlternatingItemTemplate
- ItemTemplate
- HeaderTemplate
- FooterTemplate
- SeparatorTemplate

In più il DataGrid ed il DataList supportano anche *EditItemTemplate*, che è specifico invece per uno stato particolare in cui questi due control si possono trovare, ovvero quello di modifca. Mentre nel *DataGrid*, ancora una volta, questa funzionalità è già presente e si implementa con poche righe di codice, nel DataList è necessario costruire manualmente il template, con tanto di controlli input o *TextBox*, a seconda delle presenze. Il *DataList*, rispetto agli altri, ha la possibilità di incolonnare in maniera differente i record, attraverso la proprietà *RepeatLayout*.
Consente di specificare, se *RepeatDirection* è su *Vertical* e *RepeatColumns* su 1 (i valori di default) la modalità di *rendering* del DataList. Di default questo valore è su Table, che è poi il motivo per cui viene creata una tabella con i dati estratti dalla fonte, mentre è possibile impostare questo valore su *Flow* ed utilizzare un *
* quale separatore tra ogni record. Lavorando su *RepeaterDirection* e *RepeatColumns*, invece, è possibile disporre (in verticale o orizzontale) gli items in modo tale che non siano ripetuti uno per riga, ma ce ne siano tanti quanti il valore dell'ultima proprietà appena menzionata.
Al pari degli altri controls di ASP.NET, è possibile personalizzare lo stile associato ad ogni template sfruttando le rispettive proprietà *ItemStyle, EditItemStyle*, etc, che sono tutte di tipo *TableStyle* e quindi facilmente personalizzabili.

QUALE USARE?

La risposta è semplice. Se avete bisogno di un controllo completo sull'output, di sicuro il *Repeater* è quello che fa per voi. Nel caso di design table-less o con codice XHTML, è l'unico in grado di rendere possibile questi due scenari. D'altra parte richiede che le funzionalità aggiuntive (paginazione, ordinamento, etc) vengano implementate manualmente.
Il DataList trova applicazione in quei casi in cui le funzionalità avanzate del DataGrid non servono o sono eccessive, oltre che nel caso in cui si vogliano piazzare più items per riga. Infine il DataGrid andrebbe usato, considerando il codice che produce e l'occupazione di *ViewState*, solo in intranet o comunque in ambienti ad accesso limitato.

ID	Titolo del libro	Prezzo	Note	Modifica
BU1032	Title: The Busy Executive'	19990000,0000	An overview of available database systems with emphasis on common business applications. Illustrated.	Aggiorna Annulla
BU1111	Cooking with Computers: Surreptitious Balance Sheets	11,9500	Helpful hints on how to use your electronic resources to the best advantage.	Modifica
BU2075	You Can Combat Computer Stress!	2,9900	The latest medical and psychological techniques for living with the electronic office. Easy-to-understand explanations.	Modifica
BU7832	Straight Talk About Computers	19,9900	Annotated analysis of what computers can do for you: a no-hype guide for the critical user.	Modifica
MC2222	Silicon Valley Gastronomic Treats	19,9900	Favorite recipes for quick, easy, and elegant meals.	Modifica
MC3021	The Gourmet Microwave	2,9900	Traditional French gourmet recipes adapted for modern microwave cooking.	Modifica
PC1035	But Is It User Friendly?	22,9500	A survey of software for the naive user, focusing on the 'friendliness' of each.	Modifica
PC8888	Secrets of Silicon Valley	20,0000	Muckraking reporting on the world's largest computer hardware and software manufacturers.	Modifica
PC9999	Net Etiquette		A must-read for computer conferencing.	Modifica
PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	21,5900	A must for the specialist, this book examines the difference between those who hate and fear computers and those who don't.	Modifica
PS2091	Is Anger the Enemy?	10,9500	Carefully researched study of the effects of strong emotions on the body. Metabolic charts included.	Modifica
PS2106	Life Without Fear	7,0000	New exercise, meditation, and nutritional techniques that can reduce the shock of daily interactions. Popular audience. Sample menus included, exercise video available separately.	Modifica

Fig. 4: ***Nei prossimi articoli vedremo come usare la stessa tecnica anche per inserire i dati***

CONCLUSIONI

Quale che sia il database che andrete ad utilizzare, le regole esposte in questo articolo restano le stesse. I Data Controls sono uno tra gli argomenti più estesi che ASP.NET ci consente di esplorare, poiché le personalizzazioni che si possono raggiungere sono davvero tante e tali da consertirci di adattare questi controls a tutte le nostre necessità. Ma questa è un'altra storia e l'appuntamento per parlarne è al prossimo numero!

Daniele Bochicchio

LETTURA DI XML CON IL DATASET

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace=
                    "System.Data.SqlClient" %>
<SCRIPT  language="C#" runat="server">
void Page_Load (Object sender, EventArgs e) {
  // creo il dataset
  DataSet ds= new DataSet();
  // leggo i dati dal file XML
  ds.ReadXml(Server.MapPath("feed.xml"));
  // estraggo tutti i nodi di nome item
  dg.DataSource = new DataView(
                          ds.Tables["item"]);
  dg.DataBind();
}
</SCRIPT>
<ASP:DataGrid id="dg" runat="server"/>
```

COS'È UN DATA MANAGER PROVIDER

ADO.NET è un insieme di classi incluse nel namespace *System .Data* e che si basa sul concetto di managed provider, laddove ADO invece è basato sulle tecnologie *OLE-db* o *ODBC.* Un managed provider non è altro che un insieme di classi pensate in maniera specifica per una fonte dati, nel caso specifico per un database. Ogni managed provider è composto da un numero di classi che sono implementate a partire dalle interfacce contenute nel namespace System.Data.Common e che garantiscono, per questo, un'uniformità di accesso ai dati sfruttando namespace differenti. All'interno del .NET Framework 1.1 ci sono managed provider specifici per SQL Server, Oracle ed ODBC. Chiaramente il vantaggio di avere managed provider dedicati è quello che ASP.NET riesce a dialogare in maniera più vantaggiosa con i database, perché il codice è ottimizzato e pensato per uno specifico motore. Ad esempio nel caso di Sql Server anziché usare OLE-db, ADO.NET mette a disposizione un managed provider specifico, che lavora attraverso i Tabular Data Services (TDS). Esistono poi managed provider di terze parti per MySQL, PostGre SQL ed addirittura uno per il protocollo NNTP!

Flash: all'interno della programmazione ad oggetti

Ereditarietà in ActionScript

Impariamo i principi base della programmazione ad oggetti, come estendere le classi di Action Script e facciamo qualche esempio d'uso molto utile

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
ActionScript 1.0 ed una discreta padronanza nell'utilizzo di Flash Mx e ActionScript 2.0.

Software
Flash Mx 2004 Porfessional

Impegno

Tempo di realizzazione

Una classe è un modello per la creazione di oggetti tali che abbiano in comune tutte le proprietà e i metodi in essa definiti. I metodi sono azioni compiute da un'istanza della classe. Le proprietà sono variabili definite in una classe di oggetti in cui si memorizzano dati di un determinato tipo. L'ereditarietà è la tecnica utilizzata per organizzare classi di oggetti secondo una precisa gerarchia dove una classe, generalmente indicata come classe figlia, può ereditare tutti i metodi e la proprietà di un'altra detta classe padre. Nella programmazione orientata agli oggetti con i termini interfaccia e implementazione si opera una distinzione tra cosa ci si deve aspettare dall'esecuzione di un metodo e come si è operato per implementarlo.

EREDITARIETÀ E INTERFACCE

Nel paradigma di programmazione orientato agli oggetti con il termine ereditarietà si intende una relazione formale fra due o più classi di oggetti dove la sotto classe eredita, dalla classe padre, le proprietà e i metodi in essa definiti. Dal punto di vista meramente pratico l'ereditarietà tra classi fa si che una classe possa utilizzare il codice, quindi i metodi e le proprietà, appartenente ad un'altra classe. Concettualmente però si va ben oltre, infatti, attraverso l'ereditarietà, si riesce a stabilire una vera e propria scala gerarchica tra differenti classi organizzate in gruppi. Partiamo da un semplice esempio (inheritance.fla) e da due classi definite in due documenti *.as* esterni *(Father.as* e *Daughter.as)* e vediamo come sia immediato fare ereditare alla classe *Daughter* i metodi e le proprietà della classe *Father*. Definiamo nella prima classe il costruttore e un metodo che esegue semplicemente un trace

```
class Father {
  function Father(){}
  public function testMethod(){
    trace("Ciao dalla classe Father!");}
}
```

La keyword da utilizzare per stabilire la relazione tra due classi è *extends* e va inserita subito dopo la dichiarazione del nome della classe, quindi, per creare una classe "figlia" che eredita tutti i metodi della classe "padre", usiamo questa sintassi

```
class Daughter extends Father{
  function Daughter(){}
  public function testMethodDaughter(){
  trace("Ciao dalla classe Daughter!");}
}
```

Apriamo il nostro *.fla* e, nel pannello delle azioni, scriviamo il codice necessario per creare una nuova istanza della classe *Daughter* e richiamiamo il metodo *testMethod()*

```
var a = new Daughter ();
a.testMethod();
```

Quando si invoca un metodo non presente nella classe il compilatore, prima di generare un errore dovuto alla mancanza del metodo, va a cercare il metodo nella classe con la quale è stata stabilita l'ereditarietà. Tutte le applicazioni sviluppate seguendo il paradigma della programmazione Object-Oriented possono essere viste come un diagramma formato da diverse classi legate fra loro da vincoli di ereditarietà. Nelle prime lezioni di questa guida abbiamo parlato dei tipi di dati ammissibili in Flash e, tra le novità della versione 2004, abbiamo evidenziato che ogni classe, anche quelle che abbiamo definito noi direttamente, è un data type valido. Quando definiamo un vincolo di ereditarietà tra due classi, la classe figlia viene considerata come un "sotto tipo"

della classe padre. Per questo motivo, tipizzando la variabile in cui memorizziamo la nuova istanza creata, non otteniamo errori

```
var aTyped:Father = new Daughter ();
```

Un altro aspetto molto interessante dell'ereditarietà tra classi è che, oltre a poter ereditare metodi e proprietà nella classe figlia, è possibile sfruttare questa ultima per definire delle nuove versioni dei metodi della classe padre eseguendo la cosiddetta sovra scrittura del metodo. Per creare una nuova versione del metodo *testMethod()* nella classe figlia è sufficiente aggiungere questo membro alla classe definendo le nuove operazioni da svolgere. Provate a togliere i commenti inseriti nella classe *Daughter* e ad eseguire di nuovo il codice, come sempre un esempio vale più di mille parole!

INTERFACCE

Un'interfaccia, in ActionScript 2.0, è un costrutto utilizzato per fornire una definizione di un nuovo tipo di dato. Può sembrare superfluo utilizzare un'interfaccia visto che, come abbiamo detto, già una classe definisce un nuovo tipo di dato. La differenza sostanziale è che un classe definisce il datatype e le implementazioni dei suoi metodi e delle sue proprietà, un'interfaccia lo definisce invece solo in termini astratti senza occuparsi delle implementazioni. In un'interfaccia vengono dichiarati tutti i metodi che devono essere necessariamente definiti nelle classi che la implementano ed è per questo che la si può definire come uno strumento per descrivere gli scopi per i quali viene implementata una classe. La distinzione tra implementazione del codice e descrizione dei compiti per cui è preposta una classe aiuta a mantenere più comprensibile il design dell'applicazione e di tutti i suoi componenti. Utilizziamo sempre lo stesso .fla e vediamo come definire un'interfaccia ed impostare un classe in modo tale che questa venga implementata al suo interno. Per definire un'interfaccia è necessario creare un nuovo file *.as* esterno *(TestInter .as)* che, pur rispettando le stesse regole di un file contenente una classe, invece della keyword class, inizi con la keyword interface

```
interface TestInter{function method():Void;}
```

Al suo interno troviamo solo la dichiarazione dei membri che una classe che implementa l'interfaccia deve necessariamente definire e il tipo di dati che deve essere restituito dal metodo. La keyword implements permette di specificare l'interfaccia che una classe implementa, se aprite la classe *InterfaceUsed.as* tutto sarà più chiaro

```
class InterfaceUsed implements TestInter{
  function InterfaceUsed(){}
  public function method():Void{
    trace("test!");}
}
```

Se nella classe non fosse definito il metodo *method()* il compilatore di Flash genererebbe un messaggio di errore. Gli stessi meccanismi utilizzati per stabilire l'ereditarietà fra due classi possono essere applicati alle interfacce.

ESTENDERE LE CLASSI ESISTENTI

Una delle classi più utilizzate in Flash è sicuramente la classe *MovieClip*. Abbiamo già discusso a lungo sull'importanza e le potenzialità dei clip filmato all'interno di un progetto sviluppato in Flash; vediamo ora, partendo proprio da questa classe, quanto sia facile e utile estendere con ActionScript 2.0 le classi *built-in* di Flash. Aprite un nuovo *.fla* (*extendsMovieClip.fla*) e vediamo come sia possibile aggiungere dei metodi alla classe *MovieClip* per fare in modo che questa possa muoversi via codice sullo stage. Creiamo un nuovo file *Mover.as* e dichiariamo una classe che estenda i clip filmato

```
class Mover extends MovieClip{
```

A questo punto, ragionando sulle funzionalità che questa classe dovrà avere, appare evidente che è necessario implementare un metodo pubblico che gestisca lo spostamento e tre proprietà in cui memorizziamo la velocità del movimento *(public)* e le coordinate del punto che il nostro clip filmato deve raggiungere *(private)*

```
private var toX:Number;
private var toY:Number;
public var speed:Number;
```

Ora, dopo aver definito il costruttore che assegnerà il valore 3 alla proprietà speed, implementiamo il metodo *moveToPoint()*

```
function Mover(){speed = 3;}
public function moveToPoint(x:Number, y:Number):Void{
  toX = x;
  toY = y;
  this.onEnterFrame = function(){
    _x += (toX - _x) / speed;
    _y += (toY - _y) / speed;
    if (Math.abs (toX - _x) < 0.2 && Math.abs (
                                    toY - _y) < 0.2) {
      _x = toX;
      _y = toY;
```

NOTA

TOOLS CONSIGLIATI

SePy
www.sephiroth.it

SciteFlash
http://www.bomberstudios.com/sciteflash/

JEdit
http://www.jedit.org/

Eclipse
www.eclipse.org

XCode
http://blog.pixelconsumption.com/files/Actionscript_Xcode_tutorial.rtf

```
        this.onEnterFrame = undefined;} }
}
```

Il vantaggio derivante dall'estendere la classe *MovieClip* appare quanto mai evidente dall'implementazione del metodo che sfrutta sia le proprietà *_x* e *_y* del clip filmato, sia l'evento *onEnterFrame* che tutti noi siamo abituati ad usare.
Torniamo al nostro *.fla* e vediamo come utilizzare questa classe al suo interno. Create un nuovo clip filmato contenente un cerchio, assegnategli un linkage e popolate il campo *AS2 Class* con il nome della classe che volete associare al clip filmato *(Mover)* senza l'estensione *.as*.
Aprite il pannello delle azioni e, dopo aver posizionato via codice una nuova istanza del clip filmato, chiamate il metodo *moveToPoint()* indicando le coordinate del punto che il clip filmato deve raggiungere

```
this.attachMovie("ball", "ball_mc", 0);
ball_mc.moveToPoint(300, 400);
```

Il risultato ottenuto è quello di un movimento inerziale del vostro clip filmato ed è interamente gestito da un file esterno quindi, ogni qual volta decidiate di apportare modifiche al codice, sarà sufficiente mettere mano al file *.as* esterno.

www.actionscript.it
www.risorseflash.it
www.ingenium.ws
http://www.dofactory.com/Patterns/Patterns.aspx
http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/patterns/

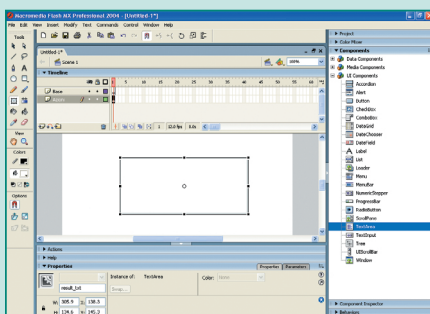
COSTRUTTORI

La programmazione ad oggetti in ActionScript 2.0 e la definizione di classi esterne è una delle novità più rilevanti introdotte in Flash Mx 2004. Abbiamo già parlato del costruttore di una classe definendolo come quella funzione che viene automaticamente richiamata dall'operatore *new* e che ci permette di eseguire delle operazioni preliminari sulle nuove istanze di una classe. Il costruttore deve avere lo stesso nome della classe e non può specificare nessun tipo di valore che venga restituito dopo la sua esecuzione. L'istruzione return può essere quindi utilizzata solo per interrompere l'esecuzione della funzione ma non per altri scopi.
Un costruttore può essere dichiarato come public o come private, ma mai come membro static della classe.
Utilizzando una dichiarazione del tipo

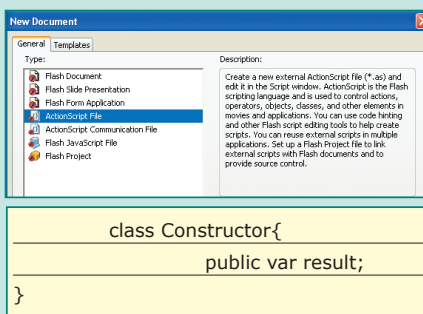
```
class Test{
  private function Test(){
  }
}
```

definiamo una classe le cui istanze non possono essere create direttamente e per questo dovremo definire un metodo della classe che ci consenta di creare nuove istanze.
L'utilizzo di questo approccio è necessario quando vogliamo simulare la definizione di classi abstract in pieno stile Java che non devono essere istanziate ma solo estese o per limitare il numero di istanze create quando la nostra applicazione è in esecuzione.
Un limite dei costruttori in ActionScript 2.0 è la mancanza di supporto per i costruttori multipli, ovvero di diverse funzioni che vengano eseguite in base al numero di argomenti passati in fase di istanziazione.
Per superare questa mancanza è possibile strutturare il costruttore della classe in modo tale che richiami diversi metodi private in base agli argomenti:

UTILIZZARE COSTRUTTORI E DISTRUTTORI

PREPARIAMO IL FLA

1 **Apriamo un nuovo fla. Nella timeline principale aggiungiamo due livelli uno per le azioni e uno contenente un campo di testo dinamico con nome *result_txt* in cui visualizzeremo i risultati delle operazioni.**

DEFINIAMO LA CLASSE CONSTRUCTOR

```
class Constructor{
      public var result;
}
```

2 **Definiamo ora la nostra classe in un file .as esterno ricordandoci che il nome del file deve essere identico al nome della classe. Per creare il file .as possiamo usare un qualunque editor di testo, oppure l'editor integrato in Flash**

DICHIARIAMOLO COME PRIVATE

```
private function Constructor(a:Number,
                              b:Number){
  if( a != undefined && b != undefined){
    newTwoArguments(a,b);
  }
  if(arguments.length == 0){
    newWithoutArguments();
  }
}
```

3 **Il costruttore della nostra classe sarà un membro private della classe e la funzione che lo definisce richiamerà due metodi differenti in base agli argomenti passati in fase di istanziazione**

```
class Test{
  private function Test(a:Number, b:Numner){
    if(arguments.length > 0){ …
    }
  }
}
```

per quanto il nostro articolo possa essere sembrato teorico, era importante introdurre i concetti che stanno alla base dell'ereditarietà delle classi.
Alcune cose potrebbero esservi sembrate astratte nello scorrere del discorso, e tuttavia se avrete la pazienza di sperimentare in pratica quanto abbiamo esposto, sicuramente ne ricaverete vantaggi inimmaginabili.

Giorgio Natil i

CHE COS'È UN DESIGN PATTERN

Quando in fase di sviluppo ci scontriamo con una problematica da risolvere, presumibilmente ciò che abbiamo riscontrato, o un problema molto simile ad esso con più di una caratteristica in comune, è stato già risolto da altri sviluppatori in situazioni analoghe o molto simili alla nostra. Un Pattern altro non è se non una formalizzazione di un modello che descrive dettagliatamente una soluzione ad uno specifico problema.
I Design Patterns sono un strumento che mette in evidenza le caratteristiche essenziali che un'applicazione deve avere per risolvere un particolare problema astraendosi dal linguaggio utilizzato e focalizzando piuttosto l'attenzione sulle classi che devono essere implementate.
Nel corso del tempo sono stati definiti diversi Design Pattern e parallelamente si è sviluppato ed evoluto un formato che viene formalmente indicato come quello da seguire per descrivere un Design Pattern.Ogni Design Pattern deve essere descritto fornendo un nome, una breve descrizione del problema che affronta, i requisiti necessari per la sua implementazione, la soluzione da adattare, le motivazioni plausibili per non scegliere di utilizzarlo e gli altri eventuali Patterns utilizzati al suo interno.
Model View Control (MVC) è uno dei Design Patterns più utilizzati.
Il modello (Model) è l'applicazione vera e propria, la vista (View) è la rappresentazione a video dell'applicazione stessa, il controllore (Controller) si occupa di sincronizzarli facendo in modo che gli input dell'utente vengano trasmessi da View a Model che a sua volta modificherà l'aspetto dell'applicazione in base alle operazioni eseguite.
MVC fondamentalmente scinde "aspetto" e "core" di una applicazione stabilendo fra loro una relazione di interdipendenza basata su una comunicazione tra istanze di oggetti.L'interfaccia grafica dell'applicazione gestita dalla classe View deve essere in grado di rispecchiare tutti i cambiamenti che avvengono nel modello in base ai cambiamenti dei dati in esso memorizzati.
Una caratteristica peculiare di questo Design Pattern, che poi altro non è altro se non uno dei principi della programmazione Orientata agli Oggetti, è la possibilità di gestire più "viste" simultaneamente anche nidificate tra loro.
Si intuisce che l'ereditarietà delle classi di cui abbiamo parlato in questo articolo, ì fondamentale per applicare la teoria dei design patterns.

BIBLIOGRAFIA

- **ACTIONSCRIPT THE DEFINITVE GUIDE FOR FLASH MX** ***Colin Moock***
- **ACTIONSCRIPT 2.0 ESSENTIALS** ***Colin Moock***
- **OBJECT-ORIENTED PROGRAMMING WITH ACTIONSCRIPT 2.0** ***Jeff Tapper***
- **AN INTRODUCTION TO OBJECT ORIENTED Programmino** ***Timothy Bud***

DEFINIAMO UN METODO STATIC

```
public static function getNewIstance(
          a:Number, b:Number):Constructor{
  return new Constructor(a,b);
}
```

4 **Per creare nuove istanze della classe dovremo definire un metodo static, ovvero un metodo che richiameremo senza dover prima creare nuove istanze della classe. Questo metodo sarà richiamato direttamente**

DEFINIAMO I METODI

```
private function newTwoArguments(
                    a:Number, b:Number):Void{
  result = arguments;
}
private function newWithoutArguments(
                                        ):Void{
  result = "No arguments";
}
```

5 **Definiamo due metodi private che vengono richiamati dal costruttore e che utilizzeremo per popolare la proprietà result precedentemente definita. Questi due metodi non fanno altro che scrivere una stringa**

UTILIZZIAMO LA CLASSE DEFINITA

```
var a = Constructor.getNewIstance(2,3);
result_txt.text = a.result;
```

6 **In Flash non faremo altro richiamare il metodo getNewIstance e popolare il campo di testo *result_txt* con il valore memorizzato nella proprietà result. Abbastanza semplice.**

Procedure e Funzioni in Visual Basic .Net

Come frazionare un programma

In questo articolo analizzeremo le procedure, le piccole unità logiche di codice che, unite, formano un'applicazione più ampia Questo modo di procedere è uno standard in programmazione

Una procedura (chiamata genericamente routine o sottoprogramma) è una parte di programma che può essere richiamata da un punto qualsiasi del codice. Il salto ad una procedura, si verifica tramite l'invocazione del nome della procedura stessa. Al termine della procedura il controllo torna al programma chiamante e l'esecuzione prosegue con l'istruzione successiva a quella che ha eseguito la chiamata. Le routine possono essere utilizzate per riunire in un unica porzione di codice operazioni ripetute e condivise (ad es. per calcoli utilizzati frequentemente).

LE ROUTINE DI VB.NET

In VB.Net si possono usare diversi tipi di routine:

- Le routine **Sub** (procedure). Una routine *Sub* è composta da una porzione di codice racchiusa tra un'istruzione *Sub* ed un'istruzione *End Sub*. Ogni volta che la routine viene chiamata, vengono eseguite le relative istruzioni a partire dall'istruzione successiva alla parola chiave *Sub* fino alla parola chiave *End Sub* (è possibile uscire da qualsiasi punto della procedura con le istruzioni *Exit Sub* o *Return*). Una routine *Sub* esegue operazioni ma non restituisce alcun valore.
- Le routine di **gestione degli eventi**. Le routine di gestione degli eventi sono routine *Sub* che vengono eseguite in risposta ad un evento generato da una azione dell'utente (ad es. il click del mouse su un bottone).
- Le routine **Function** (funzione). Una routine *Function* è simile ad una routine *Sub*, con la differenza che restituisce un valore al codice chiamante.
- Le routine **Property**, note anche come funzioni di accesso alle proprietà, consentono di gestire i valori delle proprietà di oggetti o moduli. Saranno analizzate in dettaglio nei prossimi articoli quando presenteremo la programmazione ad oggetti.

CREAZIONE DI UNA PROCEDURA

Per creare una procedura è sufficiente digitare l'intestazione della procedura nella finestra del codice e premere *INVIO*. L'intestazione di una routine può essere rappresentata semplicemente dalla parola chiave *Sub* (o *Function*) seguita da una stringa che ne rappresenti il nome.
Per esempio, possiamo scrivere

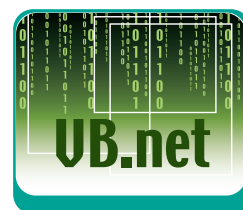
```
Sub PrimaProcedura
```

e premere INVIO, a questo punto VB riconoscerà il codice come una nuova procedura, e ne completerà il modello scrivendo automaticamente:

```
Sub PrimaProcedura ()
End Sub
```

La sintassi, in forma ridotta, di una procedura è la seguente:

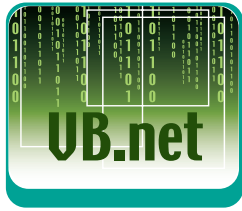
```
[{Public | Protected | Friend | Protected Friend |
                Private }] Sub nomeprocedura (argomenti)
istruzioni
End Sub
```

L'uso di una delle parole chiave (facoltative), prima dell'istruzione *Sub* determina l'area di visibilità di una procedura. In particolare una procedura dichiarata con la parola chiave *Private* può essere chiamata solo all'interno della classe, del modulo o della

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
nessuna

Software
Sistema operativo:Windows 2000/XP. Visual Basic .NET 2003

Impegno

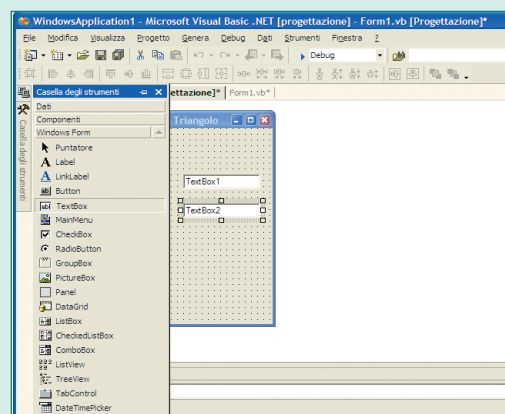
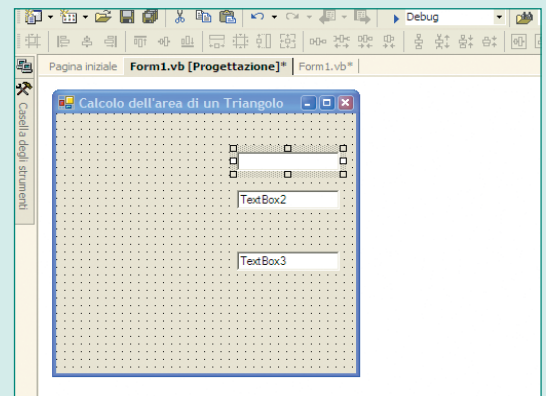
Tempo di realizzazione

form in cui viene definita, mentre una procedura dichiarata con la parola chiave *Public* è accessibile da qualsiasi parte del codice. Per default una routine è sempre *Public* perciò in questo caso è possibile omettere tale attributo, in caso contrario è sempre necessario dichiarare la visibilità di una routine con la parola chiave *Private*. Tutte le routine di gestione degli eventi sono invece *Private*. Le altre possibilità saranno più chiare quando introdurremo la programmazione ad oggetti. Come argomenti è possibile specificare un elenco di variabili (contenenti dati dell'applicazione) utilizzabili dal codice interno alla procedura. L'uso degli argomenti non è obbligatorio, ma se sono presenti, devono essere separati da una virgola e devono essere racchiusi tra parentesi.

LE FUNZIONI

Così come le procedure, le funzioni sono sottoprogrammi (routine) che possono ricevere e modificare argomenti e possono eseguire operazioni ripetute e

CALCOLA AREA

Mettiamo subito in azione le routine creando un nuovo progetto Windows Applications dal nome *CalcolaArea*. L'applicazione che andremo a realizzare, dovrà semplicemente calcolare l'area di un triangolo con le dimensioni di base ed altezza inserite dall'utente. La prima fase è quella del disegno dell'interfaccia utente.

1 **Selezioniamo la finestra *Form1* e, se non è già visualizzata, apriamo la finestra delle proprietà.**
Dalla finestra delle proprietà selezioniamo la proprietà *Name* e cambiamo subito il nome in: *FormArea*.
Selezioniamo la proprietà *Text* e modifichiamo il testo visualizzato nella barra del titolo in: *Calcolo dell'area di un Triangolo*.

2 **Selezioniamo per tre volte un controllo TextBox dalla casella degli strumenti (nella sezione Windows Form) e disegniamo i tre controlli sulla form.**

3 **Selezioniamo il primo *Textbox* e, dalla finestra delle proprietà, evidenziamo la proprietà *Name* per cambiare il nome in: *TextBoxBase*. Evidenziamo la proprietà *Text* e cambiamo il testo nella stringa vuota. Allo stesso modo: selezioniamo il secondo *TextBox* e variamo la proprietà *Name* in: *TextBoxAltezza* e la proprietà *Text* a vuoto; selezioniamo il terzo *TextBox* e variamo la proprietà *Name* in: *TextBoxArea* e *Text* a vuoto.**

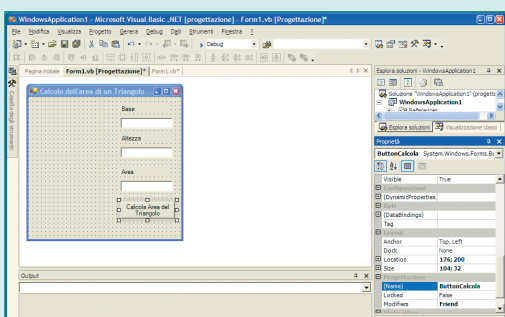
4 **Selezioniamo per tre volte un controllo Label dalla casella degli strumenti e disegniamo i tre controlli sulla form in corrispondenza dei tre TextBox disegnati in precedenza.**

L'istruzione *Exit Sub* consente di uscire immediatamente da una procedura. Per uscire da una funzione l'istruzione diventa *Exit Function*. L'esecuzione dell'applicazione continuerà con l'istruzione immediatamente successiva all'istruzione che ha richiamato la procedura o la funzione. In una procedura o funzione è possibile inserire qualsiasi numero di istruzioni *Exit Sub* o *Exit Function*. Anche l'istruzione *Return* consente di uscire subito dalla routine ottenendo lo stesso effetto di *Exit Sub*. È possibile inserire un numero illimitato di istruzioni *Return* in qualsiasi punto di una routine, nonché combinare istruzioni *Exit Sub* e *Return*.

condivise. A differenza delle procedure, le funzioni possono restituire un singolo valore che può essere assegnato ad una variabile, oppure può essere utilizzato all'interno di un'espressione.
La sintassi, in forma ridotta, di una routine *Function* è la seguente:

```
[{Public | Protected | Friend | Protected Friend |
   Private }]Function nomefunzione (argomenti) [As tipo]
istruzioni
End Function
```

5 **Selezioniamo la prima Label e, dalla finestra delle proprietà, evidenziamo la proprietà *Text* per cambiare il testo visualizzato in: *Base*. Allo stesso modo: selezioniamo la seconda Label e variamo la proprietà *Text* in: *Altezza*; selezioniamo la terza Label e variamo la proprietà *Text* in: *Area*.**

6 **Selezioniamo un controllo *Button* dalla casella degli strumenti e disegniamolo sulla form.**

7 **Selezioniamo il controllo *Button* e, dalla finestra delle proprietà, modifichiamo la proprietà *Name* in: *ButtonCalcola* e la proprietà *Text* in: *Calcola Area del Triangolo*.**

Le considerazioni sulla visibilità delle funzioni e sull'uso degli argomenti sono uguali a quelle fatte per le procedure. A differenza delle procedure è prevista l'istruzione facoltativa *As tipo*, dove *tipo* definisce il tipo di dati del valore restituito dalla funzione. Affinché una funzione possa restituire un valore, si deve usare al suo interno, una variabile con lo stesso nome della funzione ed assegnargli il valore desiderato, oppure includerlo in un'istruzione *Return*. L'istruzione *Return* consente, contemporaneamente, di assegnare il valore restituito e di uscire dalla funzione. Ad esempio le due funzioni seguenti, che restituiscono il valore di Pi Greco, hanno lo stesso effetto.

```
Public Function PiGreco() As Double
        PiGreco = 3.1415
End Function
Public Function PiGreco() As Double
        Return 3.1415
end Function
```

CHIAMATE DI PROCEDURE E FUNZIONI

Per utilizzare le istruzioni di una determinata procedura, si deve chiamare quest'ultima per nome, utilizzando uno dei due possibili metodi. Ad es. per chiamare una procedura di nome CalcolaArea:

- Call CalcolaArea() in cui il nome della procedura è preceduto dalla parola chiave Call
- CalcolaArea() specificando cioè soltanto il nome della procedura

I due metodi sono equivalenti, ma per la leggibilità del codice è importante utilizzare sempre lo stesso metodo, qualsiasi sia la vostra scelta (personalmente evito di scrivere la parola Call prima del nome della procedura). Per chiamare una routine Sub, è necessario racchiudere tra parentesi l'eventuale elenco di argomenti. Se la procedura non richiede argomenti possiamo evitare di scrivere le parentesi che saranno, in ogni caso, inserite in automatico da VB.Net. Per chiamare una funzione, è necessario scrivere il nome della funzione a destra di una qualsiasi espressione (è un espressione anche la semplice assegnazione di valore ad una variabile) così come vengono utilizzate le funzioni intrinseche di VB, quali ad es. CDbl(Numero) introdotta nell'articolo precedente. Per le funzioni vale la stessa regola delle procedure: se sono presenti degli argomenti, devono essere sempre racchiusi tra parentesi. Si può ad esempio scrivere:

```
A = AreaRettangolo(b, h)
If AreaRettangolo(b, h) < 0 Then MessageBox.Show(
                        "Errore nell'inserimento dei dati")
```

Le variabili dichiarate all'interno della routine (utilizzando l'istruzione Dim o istruzioni equivalenti) sono sempre variabili locali alla routine e la loro vita coincide con la vita della routine stessa.

• Le routine dichiarate con la parola chiave *Protected* dispongono di accesso protetto e sono accessibili solo dalla classe corrispondente oppure da una classe derivata.

• Le routine dichiarate con la parola chiave *Friend* dispongono di accesso *Friend* e sono accessibili dall'interno del programma che ne contiene la dichiarazione. Può essere chiamata da qualsiasi punto del progetto corrente, ma non dall'esterno.

• Le routine dichiarate con le parole chiave *Protected Friend* dispongono di accesso congiunto protetto e *Friend* e possono essere utilizzate dal codice in classi derivate. L'accesso *Protected Friend* può essere specificato soltanto per membri di classi.

È inoltre possibile richiamare una funzione con le stesse modalità utilizzate per le procedure:

- *Call AreaRettangolo (b,h)*
- *AreaRettangolo (b,h)*

Utilizzando questi metodi però, la chiamata di funzione non restituisce nessun valore. Nei due *TextBox* d'input (*TextBoxBase* e *TextBoxAltezza*), l'utente dovrà inserire i valori della base e dell'altezza del triangolo. Quando l'utente clicca con il mouse sul bottone, sarà lanciata la procedura di evento *ButtonCalcola_Click* che mostrerà nel *TextBox* di Output (*TextBoxArea*) il risultato dell'operazione.

In Visual Basic 6 la lunghezza dei nomi delle procedure era limitata da un massimo di 40 caratteri.

Fate attenzione al passaggio dei parametri ad una procedura, specialmente se si viene dalla scuola di VB6, non dimentichiamoci che in VB6, al contrario di VB.NET i parametri sono passati di default per riferimento (ByRef).

LE PROCEDURE DI GESTIONE DEGLI EVENTI

In VB.Net è importante distinguere due tipi di procedure:

- *Procedure generali*
- *Procedure di gestione degli eventi.*

Le procedure generali non sono legate ad alcun oggetto dell'interfaccia e, dopo essere state definite, devono essere richiamate in modo specifico dall'applicazione. Le procedure di gestione degli eventi appartengono ad una form o ad un controllo e vengono eseguite automaticamente in risposta a particolari eventi generati dall'utente (ad esempio il clic su un *Button*) o dal sistema. Nelle procedure di gestione di eventi è sempre possibile chiamare una procedura generale. Il nome delle procedure di gestione degli eventi è costituito dal nome effettivo assegnato al controllo nella proprietà *Name,* da un carattere di sottolineatura (_) e dal nome dell'evento.
Il nome della procedura associata all'evento clic del pulsante *ButtonCalcola* sarà appunto *ButtonCalcola_Click.*

ATTRIBUIRE I NOMI ALLE PROCEDURE

È possibile attribuire un nome qualsiasi ad una procedura, considerando però, i seguenti limiti:

- Il nome deve iniziare con un carattere alfabetico o con un carattere di sottolineatura (_). Se il nome di una procedura inizia con un carattere di sottolineatura, è necessario che contenga almeno un carattere alfabetico o una cifra decimale (non è possibile chiamare una procedura semplicemente _).
- Il nome deve contenere solo caratteri alfabetici, cifre decimali o caratteri di sottolineatura.
- Il nome non deve superare la lunghezza massima di 1023 caratteri.

In VB.Net qualsiasi elemento del programma può avere lo stesso nome di una parola chiave riservata, a patto di racchiuderla tra parentesi quadre ([ ]). È ad esempio possibile creare una procedura denominata While:

```
Private Sub [While]()
    'righe di codice
End Sub
```

Se nella dichiarazione della procedura non si utilizzano le parentesi quadre, VB considera l'utilizzo del nome *While* come la corrispondente parola chiave generando quindi un errore di compilazione.
Personalmente vi sconsiglio di utilizzare questa opportunità, sia perché è facile dimenticarsi di racchiudere i nomi riservati tra parentesi quadre, sia perché rende più difficile la lettura del codice. Così come abbiamo visto per i nomi delle variabili, anche per le routine vale la buona norma di utilizzare dei nomi che descrivono la funzione della procedura piuttosto che nomi generici di difficile interpretazione. È buona norma, inoltre, far precedere la routine da un breve commento sulle proprie funzionalità.

FINESTRA DEL CODICE

```
Private Sub ButtonCalcola_Click(
    ByVal sender As System.Object, ByVal e
            As System.EventArgs) Handles
                    ButtonCalcola.Click

End Sub
```

1 **Per scrivere il codice è sufficiente fare doppio click sul controllo Button. Con questa operazione si aprirà la finestra del codice con il cursore posto all'interno della procedura di evento ButtonCalcola_Click.**

DICHIARAZIONE DELLE VARIABILI

```
Dim Area As Double
Dim Base As Double
Dim Altezza As Double

Definiamo le variabili Area, Base ed
                Altezza di tipo Double
```

2 **Selezioniamo per tre volte un controllo Label dalla casella degli strumenti e disegniamo i tre controlli sulla form in corrispondenza dei tre TextBox disegnati in precedenza.**

INIZIALIZZAZIONE DELLE VARIABILI

```
Base = CDbl(TextBoxBase.Text)
Altezza = CDbl(TextBoxAltezza.Text)
```

3 **Poiché i valori immessi in un TextBox sono di tipo *String,* dobbiamo utilizzare la funzione di conversione *CDbl* che converte una qualsiasi espressione in un valore di tipo *Double*.**

PASSAGGIO DI ARGOMENTI

Spesso quando una procedura viene chiamata all'interno di un'applicazione, non richiede ulteriori informazioni per eseguire le proprie istruzioni, ma in generale, la procedura chiamata può richiedere dati all'applicazione. Per passare informazioni ad una routine vengono usate delle variabili definite argomenti. Gli argomenti possono essere passati:

- Per valore (di default)
- Per riferimento

Con il passaggio di argomenti per riferimento, la routine può modificare in maniera permanente il valore della variabile passata come argomento poiché ne viene passato l'indirizzo in memoria. Per passare un argomento per riferimento, è necessario utilizzare la parola chiave ByRef. Con il passaggio di un argomento per valore viene invece passata soltanto una copia della variabile, pertanto se la routine ne modifica il valore, tale modifica viene applicata solo alla copia e non si propaga all'intera applicazione. Per passare un argomento per valore non è necessario scrivere la parola chiave ByVal, poiché VB.Net la scriverà in automatico.

ARGOMENTI OPZIONALI

Dalla versione 4 di VB è stata introdotta la possibilità di impostare gli argomenti di una routine come opzionali o facoltativi, utilizzando la parola chiave Optional nell'elenco degli argomenti.
Gli argomenti opzionali devono essere sempre dichiarati dopo gli argomenti obbligatori perciò, quando si specifica il primo argomento opzionale, anche gli argomenti successivi devono essere opzionali, e quindi dichiarati con la parola chiave *Optional*. Per utilizzare gli argomenti opzionali si devono osservare le seguenti regole:

- Ogni argomento facoltativo deve specificare un valore predefinito.
- Ogni valore predefinito per un argomento facoltativo deve essere un'espressione costante anche se si tratta di zero o della stringa vuota.
- Ogni argomento successivo ad un argomento facoltativo deve essere anch'esso facoltativo.

La sintassi di una dichiarazione di procedura con un argomento facoltativo è la seguente:

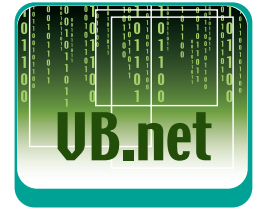
```
Sub NomeProcedura(ByVal arg1 As Tipo1, Optional
                      ByVal arg2 As tipo2 = default)
```

Quando si chiama una procedura con un argomento facoltativo, si può scegliere di non passare l'argomento. Se l'argomento opzionale non viene passato, la procedura utilizza al suo interno il valore predefinito. Se una procedura ammette più di un argomento facoltativo, è possibile scegliere di passare una qualsiasi serie di argomenti (anche vuota), l'unica regola da seguire è quella di utilizzare virgole in sequenza per contrassegnarne la posizione. Se ad esempio si vuole chiamare una procedura che ammette tre parametri opzionali passando soltanto il primo ed il terzo argomento, ma non il secondo, si deve scrivere:

```
ProceduraEsempio(arg1, , arg3)  'arg2 non viene passato
```

CONCLUSIONI

In questo articolo abbiamo accertato che è necessario suddividere un programma VB.Net in programmi più piccoli che siano più facili da scrivere, modificare e comprendere mediante le procedure e le funzioni.
Nel prossimo articolo inizieremo il viaggio nella programmazione ad oggetti.

Luigi Buono

NOTA

Un altro modo per definire una routine con argomenti facoltativi è quello di usare la tecnica dell'overload. Eseguire l'overload di una routine significa definire la stessa routine in più versioni, utilizzando lo stesso nome ma diversi elenchi di argomenti. Questo metodo diventa più oneroso con l'aumentare del numero degli argomenti facoltativi, ma ha il vantaggio di assicurare che la chiamata della procedura fornisca tutti gli argomenti facoltativi. La tecnica dell'overload sarà in ogni modo più chiara nei prossimi articoli.

LA CHIAMATA ALLA FUNZIONE

```
Area = AreaTriangolo(Base, Altezza)
TextBoxArea.Text = Area
```

4 **Poniamo la variabile *Area,* pari al valore restituito dalla funzione *AreaTriangolo* che si preoccupa di eseguire i calcoli. Infine visualizziamo nel *TextBox* corrispondente, il valore dell'area del triangolo.**

LA FUNZIONE AREATRIANGOLO

```
Public Function AreaTriangolo(ByVal Base As
        Double, ByVal Altezza As Double) As Double
      AreaTriangolo = (Base * Altezza) / 2
   End Function
```

5 **La funzione *AreaTriangolo* contiene il codice necessario per il calcolo dell'area del triangolo. Semplicemente, viene eseguito il prodotto del valore contenuto nella variabile *Base* per il valore contenuto nella variabile *Altezza* ed il risultato viene diviso per due.**

Comprendere il Document Object Model

Fondamenti di javascript

Le basi per usare il linguaggio principe per la programmazione web. Sia che stiate progettando una semplice pagina Html, sia che stiate creando un sito complesso, avrete a che fare con JavaScript

Utilizza questo spazio per le tue annotazioni

Una pagina web è visualizzata in un browser, che a sua volta è un'applicazione installata su un PC locale. I browser interpretano le pagine HTML ricevute dal server e le mostrano a video. JavaScript é un linguaggio lato client che dialoga con il Browser consentendo di modificare la visualizzazione di una pagina dopo che essa e stata ricevuta dal server, e senza dover dialogare con esso. Appare ovvio che JavaScript debba interagire con il browser accedendo a proprietà contenute nella pagina. Il DOM formalizza la struttura ad oggetti che un qualunque browser deve mettere a disposizione del programmatore e con la quale è possibile effettuare operazioni che modificano la pagina web, impostando e/o leggendo delle proprietà e richiamando metodi appartenenti a questi oggetti (come caselle di testo, tabelle, liste valori,...).
Nel corso del tempo sono stati erogati diversi standard di DOM, e alcuni browser hanno introdotto dei modelli proprietari, perciò nel programmare in JavaScript é importante affidarsi a standard ben riconosciuti e definiti. Non ci addentriamo per ora in ulteriori dettagli teorici, che analizzeremo meglio in articoli successivi. Per ora basti sapere che è stato definito un modello ad oggetti specifico per documenti HTML, sul quale il linguaggio Javascript è in grado di operare; la rappresentazione grafica del modello ad oggetti è riportata nella **Figura 1**. Il grafico mostra gli oggetti che più comunemente vengono utilizzati nella programmazione javascript. I quadratini rappresentati in scuro sono gli oggetti non ufficialmente riconosciuti dal W3C ma comunque supportati dai browser, con limitazioni che vedremo più avanti. Il primo esempio che ci accingiamo a seguire riguarda il noto elemento TextArea di HTML, che disegna una finestra di testa pronta a essere riempita da un'input.

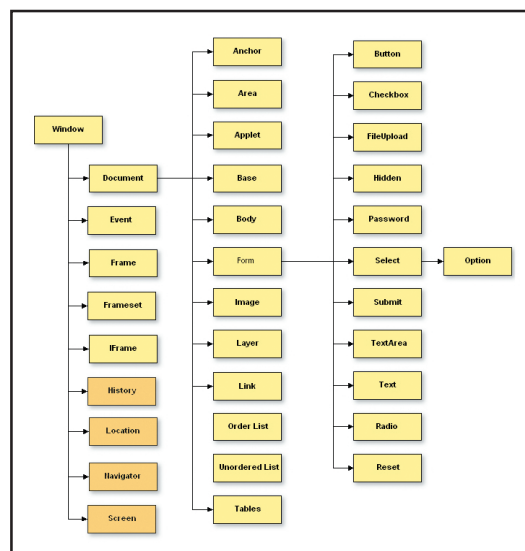



Fig. 1: ***La rappresentazione grafica del modello ad oggetti in Javascript***

HTML E GLI OGGETTI

TextArea é rappresentata da un interfaccia HTMLTextAreaElement, o viceversa potremmo dire che è HTMLTextAreaElement implementata nel browser da un oggetto che possiamo chiamare textarea (il nome dipenderà dal browser, ma non è importante nel nostro ragionamento) una cui istanza viene rappresentata con la seguente espressione all'interno di una pagina HTML:

```
<textarea name="TextArea" rows="20" cols="80">
  Io sono una area di testo a linea multipla,
  con 20 righe e 80 colonne.
</textarea>
```

Ad esempio, l''attributo cols dell'interfaccia HTML TextAreaElement farà riferimento all'attributo omologo del campo di textarea; quindi c'è da aspettarsi

che si possa mediante javascript leggere e/o modificare il valore del parametro cols della nostra textarea. In effetti e proprio così e più avanti vedremo come fare. In altri termini, la specifica del W3C relative al DOM per i documenti HTML mette in relazione l'interfaccia dell'oggetto del DOM con il tag HTML corrispondente e di questo oggetto, e quindi del tag, definisce metodi e proprietà che sono (o dovrebbero) essere supportati dai browser.
Tipicamente, una proprietà o un metodo che viene ufficialmente inserito sul DOM del W3C è implementato dai browser principali, ma è sempre consigliabile verificarlo direttamente sul sito del produttore del browser; è sempre buona norma sia effetuare un test dei propri scripts su più browser al fine di verificarne le compatibilità sia non usare metodi e proprietà specifiche di un particolare browser che in genere non sono supportate da altri browser.

INSERIMENTO DI JAVASCRIPT IN PAGINE HTML

Per prima cosa occorre vedere come sia possibile inserire del codice Javascript all'interno di una pagina HTML, quale potrebbe essere la seguente:

```
<html>
  <head>
  <title>Il mio primo programma javascript</title>
   </head>
  <body>
     <h1>Esempi di inclusione di codice Javascript</h1>
  </body>
</html>
```

Un'istruzione JS può essere inserita in una pagina HTML nei seguenti modi:

- **Primo Metodo:** mediante l'uso del tag <script> direttamente nella pagina HTML

```
<script type="text/javascript">
document.write("<b>");
   document.write("Ciao mondo! ");
   document.write("Sono il mio primo programma
                                        javascript!");
   document.write("</b>");
</script>
```

- **Secondo Metodo:** mediante l'uso del tag *<script>* che richiamo file JS esterni

```
<script src="javascript/Routines.js" type="text/
                        javascript"></script>
```

Il file Routines.js è un semplice file di testo il cui contenuto è il seguente, ed al quale potranno essrere in seguito aggiunte altre funzioni:

```
function TipoBrowser()
{document.write(window.navigator.appCodeName);}
```

- **Terzo Metodo:** All'interno di routine di eventi di oggetti HTML

```
<input type="button" value="Metodo3" name="Metod3"
  id="Ex1M3" onClick="javascript:alert('Hello world')">
```

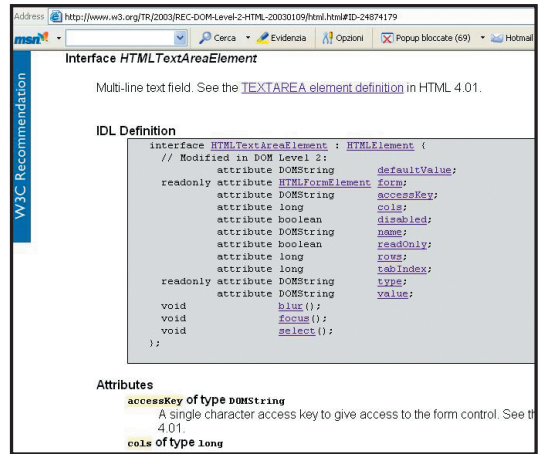



***Fig. 2:* Definizione dell'interfaccia HTMLTextAreaElement**

Il metodo migliore da utilizzare dipende dal contesto. Nel caso di brevi istruzioni può essere utilizzato il terzo metodo. In genere il secondo metodo è preferibile in quanto il codice rimane più strutturato e separato dalla parte HTML. Inoltre può essere riutilizzato su più pagine HTML, cosa che risulta molto utile nel caso di funzioni generalizzate. Il primo metodo è consigliabile nel caso in cui il codice javascript sia utilizzato solo per la pagina nella quale viene inserito. Un esempio di pagina HTML mista con funzioni JavaScript é il seguente:

```
<html>
 <head>
 <title>Il mio primo programma javascript</title>
 <!--  Secondo Metodo -->
 <script src="javascript/Routines.js"
                          type="text/javascript" >
 </script>
 </head>
  <body>
  <h1>Esempi di inclusione di codice Javascript</h1>
  <p>
           Il primo metodo restituisce il classico 'Hello
                               World' in grassetto....
    </p>
   <!-- Primo Metodo -->
         <script type="text/javascript">
         document.write("<b>");
```

Nella progettazione di applicazioni web, i linguaggi che vengono utilizzati per lo sviluppo di tali applicazioni sono sostanzialmente di due tipi:

∑ Linguaggi lato Client: tali linguaggi, tra i quali possiamo annoverare VisualBasic Script e Javascript, permettono di interagire principalmente con gli elementi HTML componenti la pagina senza dover scambiare i dati con il server. Questo permette di ridurre drasticamente i tempi di esecuzione delle istruzioni, ma rende necessario il dover conoscere il tipo di browser sul quale questi script verranno eseguiti.

∑ Linguaggi lato Server: tali linguaggi, tra i quali possiamo annoverare PHP, ASP, Coldfusion, JSP, Perl, C CGI, etc vengono interpretati dal server, che esegue le istruzioni e presenta i dati. E' compito dell'utente manipolare i dati dinamici e presentarli sotto forma di codice HTML.

Alcune operazioni, quali ad esempio l'accesso a database o scrittura di file, è possibile effettuarle solo avvalendosi di linguaggi di scripting lato server, mentre per le interazioni con gli elementi della pagina è sicuramente più conveniente utilizzare linguaggi lato client.

```
        document.write("Ciao mondo! ");
        document.write("Sono il mio primo programma
                                    javascript!");
        document.write("</b>");
    </script>
  <p>
```

Il secondo metodo richiama la funzione TipoBrowser() definita nel file Routines.js della cartella 'javascript'. Cliccando sul pulsante visualizzato viene richiamata la funzione in questione.

```
  </p>
  <!-- Secondo metodo -->
      <form name="frmTerzoMetodo" id="frmTerzoMetodo">
        <input type="button" value="Metodo2" name=
      "Metodo2" id="Ex1M2" onClick="TipoBrowser()">
  </form>
  <p>
```

Il terzo metodo visualizza un messaggio di popup. Allo stesso modo dell'esempio precedente sarà necessario cliccare sul pulsante per richiamare la funzione corretta

```
</p>
<!-- Terzo metodo -->
   <form name="frmTerzoMetodo" id="frmTerzoMetodo">
     <input type="button" value="Metodo3" name=
         "Metod3" id="Ex1M3" onClick="javascript:alert
                                  ('Hello world')">
  </form>
</body>
</html>
```

Nell'intestazione, all'interno del tag <head> viene inserito il tag script nel quale sono impostati gli attributi:

- src che punta al file javascript (con estensione js) posizionato nella cartella denominata "javascript" relativamente alla posizione del file HTML
- type che indica il tipo di file puntato da src il cui valore "text/javascript" sta ad indicare che il file è di testo e che contiene istruzioni javascript.

In questo modo viene applicato il secondo metodo di inclusione. Scendendo verso il basso all'interno di un altro tag *<script>* troviamo del codice javascript composto da una serie di istruzioni: *document.write ("Una stringa di testo...")*. Si noti innanzi tutto che le stringhe in parentesi possono essere sia stringhe di testo che tag HTML (nel nostro esempio il tag per rendere il grassetto *<b>*). Questo è un primo esempio di applicazione del modello del DOM. La classe *document* che implementa l'interfaccia HTMLDocument, che deriva dalla classe padre window, secondo le specifiche del W3C ha un metodo denominato write(in DOMString text), il cui scopo è quello di scrivere sul documento (ossia la pagina HTML) la stringa text che gli viene passata come parametro. In definitiva, con questa istruzione si può scrivere HTML all'interno della pagina del documento. Questo rappresenta un esempio di inclusione di codice javascript all'interno della pagina HTML. Proseguendo ancora con il codice abbiamo un esempio di funzione javascript richiamato dall'interno di un tag HTML. In particolare, la funzione *TipoBrowser()* definita nel file esterno *Routines.js* viene richiamata in seguito alla pressione del pulsante etichettato con *'Metodo2'*, che scatena l'evento *onClick* del pulsante stesso. L'ultimo esempio è simile al primo, con la differenza che invece di richiamare una funzione codificata in un file esterno, la funzione è scritta direttamente all'interno del tag. Il codice *onClick="javascript:alert('Hello world')* significa "sul click del pulsante esegui il codice alert('Hello world') che è scritto in linguaggio javascript". Attenzione che scrivendo semplicemente *onClick= "alert('Hello world')* la cosa non funzionerebbe. Occorre sempre dire il tipo di linguaggio di scripting che si utilizza. La funzione javascript alert("Messaggio") invia una messaggio di popup con la stringa che gli si passa in input (l'equivalente della funzione MsgBox di Visual Basic 6). Si noti inoltre come il flusso del linguaggio sia sequenziale e segua esattamente quello della creazione della pagina HTML, partendo dall'inizio della pagina e proseguendo sino alla fine. Per finire non bisogna dimenticare che le nostre pagine web possono essere visualizzate su browser che non hanno supporto di scripting o per i quali tale supporto è stato disabilitato. In tal caso viene in supporto il tag <noscript>. L'utilizzo del tag è il seguente:

```
<script type="text/javascript" >
  document.write("<b>Questa frase viene scritta solo
             se il browser supporta javascript</b>");
</script><noscript>
  <b>Questa frase viene scritta solo se il browser non
                          supporta gli script</b>
</noscript>
```

Per browser un po' datati, si rende necessario inserire il codice javascript tra commenti di tipo HTML posti all'interno del tag <script>, in modo tale che il browser li salti e non li mostri sulla pagina. Un esempio potrebbe essere il seguente:

```
<script type="text/javascript" >
  <!--
    document.write("<b>Istruzione commentata per
              compatibilità con browser datati</b>");
  -->
</script>
```

Sempre per browser datati è consigliabile specificare la versione del linguaggio javascript che si sta utilizzando, ad esempio:

```
<script type="text/javascript" language="javascript1.2">
  <!--
    document.write("<b>Viene specificata la versione
                  di javascript per browser datati</b>");
  -->
</script>
```

utilizzando l'opzione language del tag <script>. Così facendo, ci si garantisce che il vostro codice venga interpretato correttamente da browser "vecchio stile", senza per altro creare problemi ai browser più modermi. Se la versione non viene specificata, il browsere usa l'ultima versione di javascript che ha disponibile. L'ultima versione disponibile di javascript è la 1.6, supportata solo da alcuni browser. Leggete il box relativo alla versione di javascript per ulteriori simpatici approfondimenti sulla questione. Personalmente consiglio comunque sempre di testare il software su differenti browser che pensiate possano essere utilizzati dalla utenza della vostra applicazione web.

ED ORA UN PO' DI CODICE...

...che non fa mai male. Prendete il vostro editor di testo preferito e copiate il codice che segue (che potete trovare anche nel software messo a disposizione con l'articolo):

```
<html>
<head>
  <title>Javascript - Articolo 1 - parametri del
                                    browser</title>
</head>
<body>
<h1>Quello che avreste sempre voluto sapere del
                              vostro browser...</h1>
<script type="text/javascript">
function TipoBrowser()
{  document.write("<i>Nome del browser: </i>");
   document.write(window.navigator.appName);
   document.write("<br>");
   document.write("<i>Versione del browser: </i>");
   document.write(window.navigator.appVersion);
   document.write("<br>");
   document.write("<i>Sotto-versione del browser: </i>");
   document.write(window.navigator.appMinorVersion);
   document.write("<br>");
   document.write("<i>Code name del browser: </i>");
   document.write(window.navigator.appCodeName);
   document.write("<br>");
   document.write("<i>Linguaggio del browser: </i>");
   document.write(window.navigator.browserLanguage);
   document.write("<br>");
   document.write("<i>I cookie sono abilitati o no ?: </i>");
   document.write(window.navigator.cookieEnabled);
   document.write("<br>");
   document.write("<i>Stai lavorando in modalità
                          on-line oppure no?: </i>");
   document.write(window.navigator.onLine);
   document.write("<br>");
   document.write("<i>Il vostro sistema operativo: </i>");
   document.write(window.navigator.platform);
   document.write("<br>");
   document.write("<i>La lingua del vostro sistema
                                   operativo: </i>");
   document.write(window.navigator.systemLanguage);
   document.write("<br>");
   document.write("<i>La classe della CPU ?: </i>");
   document.write(window.navigator.cpuClass);
   document.write("<br>");
   document.write("<i>Intestazione dello user-agent
                      inviato dal vostro browser: </i>");
   document.write(window.navigator.userAgent);
   document.write("<br>");
   document.write("<i>Intestazione della lingua dello
            user-agent inviato dal vostro browser: </i>");
   document.write(window.navigator.userLanguage); }
TipoBrowser();
</script>
</body>
</html>
```

Salvate il file come Browser.html ed aprite il file nel vostro browser web; potrete visualizzare un po' di informazioni sul vostro browser. Nel codice ci sono molti degli elementi di javascript che utilizzeremo nel seguito. Ovviamente, l'inserimento di codice javascript in una pagina HTML (si è scelto per semplicità il primo metodo). Viene poi definita una funzione javascript denominata TipoBrowser() che viene richiamata subito dopo, all'interno dello stesso tag script. Non ci soffermeremo oltre sul significato delle proprietà del browser richiamate nello script, del resto immediatamente comprensibili e sulle quale discuteremo in dettaglio nel proseguo degli articoli.

CONCLUSIONI

In questa puntata avete imparato le basi del linguaggio javascript ed avete potuto creare un esempio di programma (che non è il solito 'Hello World') che vuole evidenziare anche un po' le potenzialità del linguaggio.
Nelle prossime puntate ci addentreremo nella sintassi del linguaggio.
L'avventura è appena iniziata…

Danilo Fadda

NOTA

All'interno del tag <script> viene utilizzato sovente l'attributo language, che viene utilizzato dai vecchi browser sia per capire quale versione di javascript viene utilizzata nel codice contenuto all'interno del tag, sia per capire se tale versione viene supportata dal browser stesso. Ad esempio:

```
<script language=
            "javascript1.2">
....codice javascript...
</script>
```

Tale attributo non viene utilizzato all'interno degli script dell'articolo, in quanto deprecato da W3C a favore dell'attributo type del tag <script>. In articoli successivi approfondiremo l'argomento, non banale, relativo alla versione di javascript.

Symbian primi passi

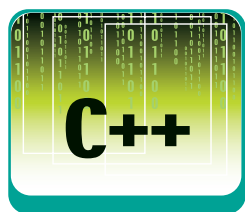
In questo articolo e nei successivi impareremo come programmare applicazioni per il sistema operativo Symbian usando C++ Utilizzare C++ ci garantisce massima flessibilità e portabilità

L'elevato uso dei dispositivi mobili ha portato negli ultimi anni ad un'evoluzione delle piattaforme hardware e software per strumenti come cellulari, notebook e palmari. Il telefonino in particolare è diventato uno strumento di primaria importanza nella vita di tutti i giorni e l'utente vuole avere a portata di mano tutto ciò che offre un PC tradizionale. Symbian OS è una tra le principali tecnologie moderne che rendono il cellulare uno strumento completo. Durante il corso conosceremo meglio la piattaforma e impareremo a sfruttarne le funzionalità.

Fig. 1: ***È importante settare variabili d'ambiente***

INSTALLIAMO GLI STRUMENTI

Lo strumento principale per un programmatore è l'SDK (*Software Development Kit*). È necessario quindi procurarsi l'SDK relativo al cellulare su cui si vuole scrivere l'applicazione. Ogni azienda che produce cellulari Symbian OS mette a disposizione il proprio SDK, che può anche riferirsi a più modelli. All'indirizzo *http://www.forum.nokia.com/main/0, 6566,034-4,00.html* si possono trovare i kit per i cellulari Nokia. Scriveremo software per il Nokia 7650, usando l'SDK 1.2, ma con qualche piccola modifica è possibile sfruttare lo stesso codice per il Nokia 6600 usando l'SDK 2.0.
Un elenco completo dei cellulari Symbian è reperibile all'url *http://www.symbian.com/phones/index.html*. Nella fase di build dell'applicazione è indispensabile un interprete Perl. *ActivePerl* fa al caso nostro (*http:// www.activeperl .com/Products/ActivePerl/*), ricordandoci di settare opportunamente le variabili d'ambiente (**Figura 1**).
Abbiamo inoltre bisogno di Microsoft Visual C++ (*http://msdn.microsoft.com/visualc/*).

CREAZIONE DEL PROGETTO

Una semplice applicazione in Symbian richiede la creazione di diversi file, siano essi sorgenti o risorse. Il software ApplicationWizard di casa Nokia ci viene in aiuto. Con l'SDK 1.2 è necessario aggiungere il template di creazione direttamente in Visual C++. Copiamo quindi i file presenti nella cartella *\Symbian\6.1\Series60\Series60Tools\applicationwizard*

Fig. 2: ***Settiamo il nome dell'applicazione nel wizard lasciando inalterato il resto***

nel path dei template *Microsoft Visual Studio\Common\MsDev98\Template*. Avviamo Visual C++ e dal menu *"File"* selezioniamo *"New"*. Nella tab dei progetti troviamo adesso *"Series 60 AppWizard"*. Selezioniamolo e settiamo nome e path del progetto, nel nostro esempio *c:\HelloWorld*. Per la creazione di un nuovo progetto è necessario specificare solo il titolo dell'applicazione, lasciando invariato tutto il resto (**Figura 2**). Nella directory *c:\HelloWorld\src* sono stati creati i sorgenti di un programma base. Per la comprensione del codice diamo uno sguardo al linguaggio utilizzato.

C++ PER SYMBIAN OS

Sebbene C++ sia il linguaggio utilizzato per Symbian OS, esistono alcune differenze e convenzioni particolari sui nomi. I nomi delle classi cominciano con una delle lettere C, T, R o M, con i seguenti significati.

- **C:** Classe allocata in memoria derivata da CBase
- **T:** Classi che non contengono nessun oggetto esterno
- **R:** Classi che contengono collegamenti a risorse reali
- **M:** Classi di interfaccia

In Java ogni classe discende indirettamente da Object, in C++ per Symbian ogni classe discende da *CBase*. La creazione di un nuovo oggetto si effettua con il seguente costrutto:

```
CMiaClasse mioOggetto = new (ELeave) CMiaClasse();
```

"ELeave" introduce il discorso delle eccezioni. In C++ e in Java è previsto un costrutto *try-catch* per la cattura delle eccezioni, facendo scorrere l'errore tramite un oggetto. In C++ per Symbian si parla più propriamente di *"leave"*. Non viene più gettata un'eccezione ma effettuato un *leave*. La funzione *User::Leave()* è l'analogo del *throw*. Il *leave* produce un codice d'errore di 32 bit anziché un oggetto, rendendo meno pesante la gestione. Un *"panic"* è un errore non catturabile e causa la terminazione del processo. La funzione relativa è *User::Panic()*. I *leave* possono causare spreco di memoria e di risorse. Vediamo il seguente codice:

```
void mioMetodo() {
  CMiaClasse * mioOggetto = new (ELeave) CMiaClasse();
  miaClasse -> metodoConLeaveL();
  delete miaClasse;
}
```

Se *metodoConLeaveL()* si interrompe per un errore, *mioMetodo()* non termina la sua esecuzione e delete *miaClasse* non viene effettuato. Perdendo il riferimento all'oggetto non vi è alcuna possibilità di liberare memoria. In un dispositivo portatile, dove le risorse sono scarse, ciò è importante!
Cleanup Support è la soluzione al problema.

CLEANUP SUPPORT

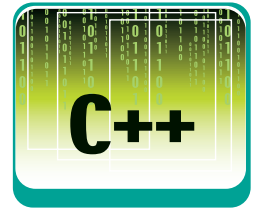
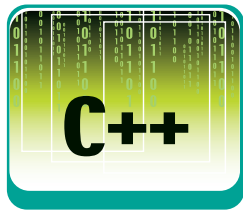
Al verificarsi di un errore il framework Symbian libera la memoria occupata da tutti gli oggetti contenuti nello stack *"Cleanup"*. La seguente gestione:

```
void mioMetodo() {
  CMiaClasse * mioOggetto = new (ELeave) CMiaClasse();
  CleanupStack::PushL(miaClasse);
  miaClasse -> metodoConLeaveL();
  CleanupStack::Pop();
  delete miaClasse;
}
```

Fig. 3: ***Un'immagine evocativa dei telefoni Symbian OS in commercio***

permette di tenere traccia del riferimento all'oggetto. Se *metodoConLeaveL()* esce a causa di un errore, la memoria di *mioOggetto* viene comunque liberata. Un'ultima importante considerazione riguarda il *leave* all'interno di un costruttore. Il costruttore di una classe viene richiamato solo dopo che è stata assegnata la memoria per il nuovo oggetto. Un *"leave"* in fase di costruzione porterebbe ad uno spreco. In questo caso viene utilizzata una tecnica chiamata *"Costruzione in 2 fasi"*. Ecco un esempio:

```
class CMiaClasse {
  CAltraClasse * nuovoOggetto;
  CMiaClasse() {}
  ConstructL() {
    // L'oggetto viene istanziato qui anziché nel
    // costruttore
    nuovoOggetto = new (ELeave) CAltraClasse(); }
}
CMiaClasse* oggetto = new (ELeave) CMiaClasse();
cleanupStack::PushL(oggetto);
oggetto -> ConstructL();
cleanupStack::Pop();
```

Il costruttore non si preoccupa di gestire situazioni potenzialmente rischiose, lasciando il compito ad

Utilizza questo spazio per le tue annotazioni

una funzione secondaria che per convenzione è chiamata *ConstructL()*, sulla quale è possibile ricorrere al *Cleanup Support*. Un utile approccio consiste nell'utilizzare due metodi statici per effettuare la costruzione in 2 fasi:

```
static CMiaClasse * CMiaClasse::NewLC() {
  CMiaClasse * questo = new (ELeave) CMiaClasse();
  CleanupStack::PushL(questo);
  questo->ConstructL();
  return questo; }
static CMiaClasse * CMiaClasse::NewL() {
  CMiaClasse * oggetto = NewLC();
  CleanupStack::Pop();
  return oggetto;
}
```

La funzione *NewL()* permette di ottenere un'istanza della classe. La lettera *"L"* finale rappresenta per convenzione la possibilità che essa effettui un leave, e la lettera *"C"* implica che l'espulsione dallo stack è a carico del metodo chiamante.

HELLOWORLD

Per la stesura di HelloWorld utilizziamo lo scheletro dei sorgenti creati precedentemente con Visual C++. Apriamo il file *c:\HelloWorld\src\HelloWorldAppUi.cpp* e, nella funzione *HandleCommandL()*, assicuriamoci che il blocco *"case EhelloworldCmdAppTest:"* contenga le seguenti righe:

```
_LIT(message,"Hello World!");
CAknConfirmationNote* note = new (Eleave)
                                CAknConfirmationNote();
note->ExecuteLD(message);
```

Inoltre aggiungiamo:

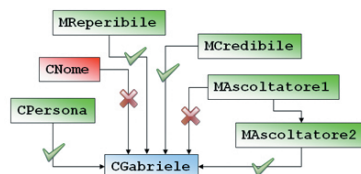
```
#include <aknnotewrappers.h>
```

La descrizione dettagliata del framework grafico verrà trattata nel successivo articolo. È importante adesso concentrarsi sulla procedura di build, che permette l'esecuzione del software sul nostro cellulare.

EREDITARIETÀ MULTIPLA

In C++ per Symbian l'ereditarietà multipla è stata sostituita da semplici regole. Tali regole riprendono la comoda sintassi Java.

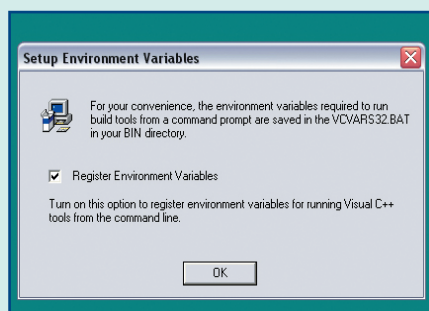
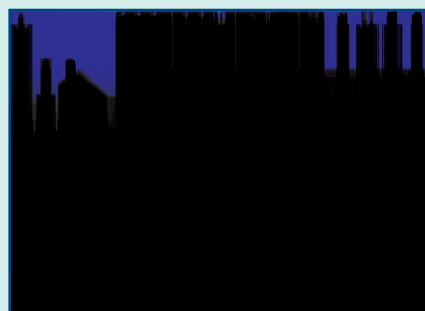
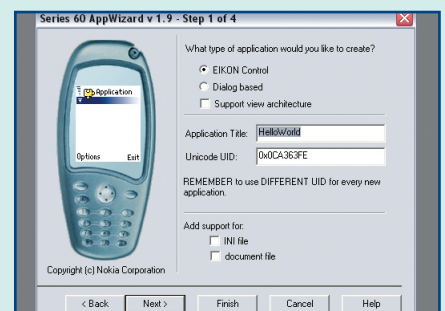
Classe C: Classe (concetto classico)
Classe M: Interfaccia

- **Possiamo derivare una classe C da una sola classe C e zero o più classi M.**
- **La derivazione dalla classe C deve essere dichiarata per prima.**
- **Non possiamo derivare una classe C da due classi M contemporaneamente se la prima è derivata dalla seconda.**

L'ULTIMA FASE: IL BUILD

Prima di installare il software sul telefonino è necessario ovviamente compilarlo. Il file *c:\HelloWorld\group\helloworld.mmp* contiene le specifiche di progetto. All'interno troviamo una riga *SOURCE* per ogni file sorgente:

```
SOURCE  SourceFile.cpp
```

Apriamo una console testuale e portiamoci nella directory *c:\HelloWorld\group*.
Lanciamo i due comandi:

CREIAMO LA PRIMA APPLICAZIONE

1 **Installiamo gli strumenti che ci servono: Nokia SDK 1.2, Microsoft Visual C++ 6.0 e ActivePerl. Durante l'installazione di Visual C++ ricordiamoci di settare le variabili d'ambiente come mostrato in figura. È possibile farlo anche tramite VCVars32.BAT.**

2 **Aggiungiamo a Visual C++ il template Nokia per la creazione assistita di un'applicazione base. Copiamo *avkonappwiz.awx* e *avkonappwiz.hlp* da *Symbian\6.1\Series60\Series60Tools\applicationwizard* a *Microsoft Visual Studio\Common\MSDev98\Template***

3 **Apriamo Visual C++ e clicchiamo su *File*, poi su *New*. Dalla tab *projects* selezioniamo *Series 60 AppWizard*, digitiamo il nome del progetto, settiamo la directory e premiamo *OK*. Nel nuovo dialog inseriamo il nome dell'applicazione e clicchiamo su *Finish*.**

```
bldmake bldfiles (crea il file abld.bat)
abld build thumb urel (compilazione e link)
```

Abbiamo così creato il programma *HelloWorld.app* e alcuni file necessari alla sua esecuzione. Tali file risiedono in *\Symbian\6.1\Series60\Epoc32\release\thumb\urel*.
Per l'installazione sul cellulare abbiamo due possibilità. È abbastanza scomodo copiare tutti i file nella directory *c:\system\apps\helloworld* del dispositivo mobile, perciò decidiamo di creare un file installativo *helloworld.sis* che farà il lavoro per noi. I file *.sis* sono riconosciuti dai cellulari Symbian come software autoinstallanti. Per creare un file *.sis* è necessario creare un file di testo *.pkg* che contiene: nome dell'applicazione, numeri di versione, parametri di compatibilità e file da installare. Nel nostro esempio Visual C++ l'ha già creato per noi. In console, posizioniamoci nella directory *c:\HelloWorld\install* e digitiamo:

```
makesis helloworld.pkg
```

Se tutto va a buon fine nella stessa directory troviamo *helloworld.sis*. Installandolo sul cellulare ed eseguendolo otteniamo le schermate in **Figura 4**.

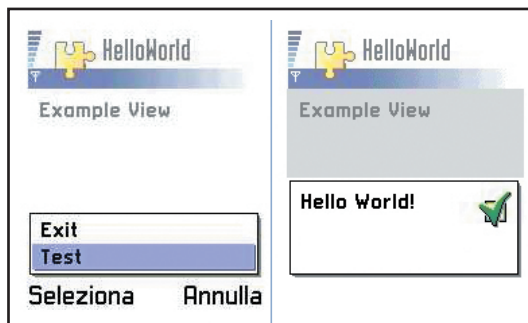
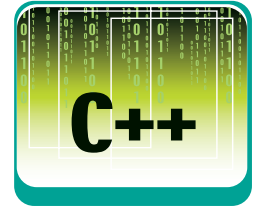



Fig. 4: *Un altro esempio dell'applicazione in esecuzione*

L'aggiunta di ulteriori file al pacchetto comporta la modifica di *helloworld.pkg*, che presenta una sintassi banale. Un ottimo strumento per la creazione dei file *.sis* è Sisar, distribuito insieme a qualunque SDK e richiamabile dal menu *START*.

CONCLUSIONI

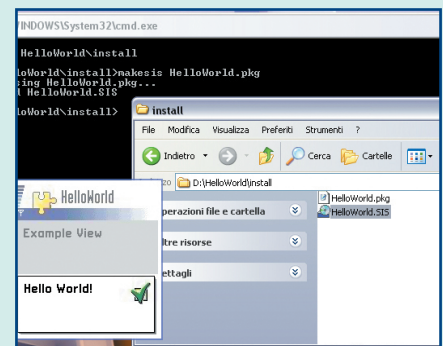
Abbiamo visto come creare una semplice applicazione per il Nokia 7650 sfruttando l'SDK 1.2 fornito dalla Nokia. Più in particolare è stato necessario aggiungere alcune istruzioni in punti ben definiti di un progetto base creato con appositi strumenti. Sebbene sia necessario puntare ad una specifica piattaforma di sviluppo, le conoscenze acquisite nell'articolo ci permettono anche una migrazione a diversi o a nuovi dispositivi portatili basati su Symbian OS, il tutto con "poco" sforzo.
Non è tuttavia da escludere la possibilità che il codice da noi scritto diventi "inutile", se pensiamo alla facilità con cui al giorno d'oggi sostituiamo il cellulare con uno nuovo. La chiave di tutto rimane comunque il continuo aggiornamento.

Antonio Trapani

C++ VS JAVA

Sebbene Java sia un linguaggio a oggetti flessibile e affidabile, esonerando il programmatore dal gestire la memoria, esso presenta delle debolezze. A differenza dei PC, dove la potenza di calcolo riduce la differenza di prestazione di un programma scritto in C++ e di uno scritto in Java, i dispositivi portatili hanno risorse limitate. Su un cellulare, Java è più "pesante". Al giorno d'oggi inoltre le JVM non offrono il pieno controllo dei dispositivi, mancando ad esempio di funzionalità importanti come l'accesso al filesystem o l'esecuzione di metodi nativi. L'uso del C++, grazie alla piattaforma Symbian, permette un controllo diretto su tutto il sistema.

4 **È il momento di aggiungere il nostro codice. Apriamo *c:\HelloWorld\src\HelloWorldAppUi.cpp* e all'interno della funzione *HandleCommandL()* inseriamo le istruzioni riportate nell'articolo.**

5 **Apriamo una console dei comandi e posizionamoci nella directory *group* appartenente al nostro progetto. Digitiamo *bldmake bldfiles* e subito dopo *abld build thumb urel*. Il programma è compilato e pronto per essere trasferito.**

6 **Per creare il file d'installazione portiamoci invece nella directory install e digitiamo *makesis HelloWorld.pkg*. Trasferiamo *HelloWorld.sis* sul telefonino, installiamolo e lanciamolo. Se tutto è andato a buon fine vedremo la finestra di *HelloWorld*.**

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

JAVA

INTERROGARE LO STATO DELLE PORTE

Ecco un semplice tip per interrogare lo stato delle nostre porte(seriali e parallele). Il tip fa uso del package *commapi* scaricabile dal sito della sun. Il programma innanzitutto crea la lista delle porte installate sulla nostra macchina e tramite l' enumerare invocando su ogni porta il metodo open per testarne la disponibilità,se tale metodo solleva l'eccezione *"javax.comm.PortInUseException"* allora significa che la porta in esame è occupata da qualche altra applicazione.

Tip fornito dal sig. Avolio Antonio

```
package comunicazioneporte;
import java.util.*;
import javax.comm.*; //package contenente le commapi
public class scansionePorte {
  public static void main(String[] args) {
    //Creazione della lista delle porte installate sulla macchina
    Enumeration porteInstallate =
                                    CommPortIdentifier.getPortIdentifiers();
    //Oggetto che interaggisce con i driver
    CommPortIdentifier portaID=null;
    //Enumerazione delle singole porte
    while (porteInstallate.hasMoreElements()) {
      try {
         portaID = (CommPortIdentifier) porteInstallate.nextElement();
         System.out.print(portaID.getName());
         //verifica se si tratta della porta seriale
         if (portaID.getPortType() == 1) {
            System.out.print(" SERIALE");
            portaID.open("prova", 100);
       }
       //verifica se si tratta della porta parallela
       if (portaID.getPortType() == 2) {
           System.out.print(" PARALLELA");
           portaID.open("prova", 100);
       }
           System.out.print(" Porta libera");
      }
      catch (javax.comm.PortInUseException e) {
           if((e.currentOwner).equals("Unknown Windows Application"))
           System.out.print(" La porta "+portaID.getName()+" e'
                          occupata da un'applicazione di windows");
           else
              System.out.print(" La porta "+portaID.getName()+
                                              " "+e.currentOwner); }
         System.out.println();
     } //while
   }
}
```

VB.NET

INVIARE E-MAIL CON .NET

La soluzione che vi propongo permette di inviare mail grazie alle utilissime classi del framework .NET. Ma la novità sta nel fatto che ho aggiunto le istruzioni utili all'autenticazione verso il server SMTP che solitamente non sono documentate negli esempi che si trovano in rete. Utile per chi ha un sito in hosting ma non un server SMTP disponibile presso il proprio provider. Nell'esempio ho utilizzato il server di Spymac

Tip fornito dal sig. Gianluca Negrelli

```
// classe per l'invio di mail attraverso un server SMTP
// con autenticazione di user name e password
// linguaggio c#
// framework .NET 1.1
// 26/01/05
// autore Gianluca Negrelli
using System.Web.Mail;
namespace gn.utility
{
  public class classMail
  {
    public static bool InviaMail(string sBody, string sFrom, string
                                              sTo, string sSubject)
```

```
  {
    try
    {
      // tentativo di invio mail con autenticazione
      // il server è spymac
      MailMessage MyMsg=new MailMessage();
      MyMsg.Body=sBody;
      MyMsg.From=sFrom;
      MyMsg.To=sTo;
      MyMsg.Subject=sSubject;
      // tentativo di autenticazione verso il server spymac
      MyMsg.Fields.Add("http://schemas.microsoft.com/cdo/
       configuration/smtpauthenticate","1"); //basic authentication
      MyMsg.Fields.Add("http://schemas.microsoft.com/cdo/
          configuration/sendusername", "xxxxxx@spymac.com");
                                                      //username
      MyMsg.Fields.Add("http://schemas.microsoft.com/cdo/
            configuration/sendpassword", "xyz"); //password
      // tentativo di autenticazione verso il server spymac
       SmtpMail.SmtpServer="mail.spymac.com";
       SmtpMail.Send(MyMsg);
       // comunica il buon fine
       return true;
    }
    catch
    {
      // comunica l'errore
      return false;
    }
  }
 }
}
```

IL TIP DEL MESE

FATE VIBRARE LA VOSTRA JFRAME

Il codice completo per far vibrare una *JFrame* (o qualsiasi altro *JComponent*). Per velocizzare il tremolio basta cambiare il 50 della funzione trema() [che sarebbero i millisecondi di occorrenza del timer]: cambiandolo in 100 andrà più lento; in 25 andrà più veloce. Per far durare il tremolio più tempo basta aggiungere istruzioni case alla funzione *spostaFinestra()* e cambiare il 20 nell'istruzione *if (numTremate == 20)* con il numero di istruzioni case inserite.

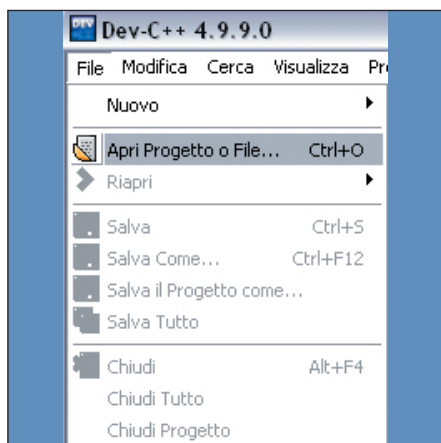
Tip fornito dal sig. Rosario Capparelli

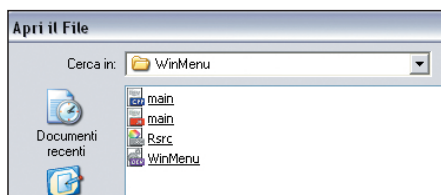
```
package trema;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.Timer;
public class Trema extends JFrame{
  Timer t;
  private int numTremate;
  private Point p;
  public Trema(){
  this.setDefaultCloseOperation(
                              EXIT_ON_CLOSE);
  JButton bottone = new JButton("trema");
  bottone.addActionListener(new
                             ActionListener(){
    public void actionPerformed(
                           ActionEvent arg0) {
      trema();
  }});
  this.getContentPane().add(bottone);
  this.pack();
  this.setLocationRelativeTo(null);
  this.setVisible(true);
}
void trema(){
  t = new Timer(50,new ActionListener(){
    public void actionPerformed(
                          ActionEvent event){
      spostaFinestra();
  }});
  t.start();
}
protected void spostaFinestra() {
  numTremate = numTremate + 1;
  if(numTremate == 20){
    stopTimer();
    numTremate = 0;
  }
  else{
    p = this.getLocation();
    switch(numTremate){
      case 1: this.setLocation(p.x - 10,p.y
                                    );break;
      case 2: this.setLocation(p.x + 20,p.y
                                    );break;
      case 3: this.setLocation(p.x - 20,p.y
                                    );break;
      case 4: this.setLocation(p.x + 20,p.y
                                    );break;
      case 5: this.setLocation(p.x - 20,p.y
                                    );break;
      case 6: this.setLocation(p.x + 20,p.y
                                    );break;
      case 7: this.setLocation(p.x - 20,p.y
                                    );break;
      case 8: this.setLocation(p.x + 20,p.y
                                    );break;
      case 9: this.setLocation(p.x - 20,p.y
                                    );break;
      case 10:this.setLocation(p.x + 20,p.y
                                    );break;
      case 11:this.setLocation(p.x - 20,p.y
                                    );break;
      case 12:this.setLocation(p.x + 20,p.y
                                    );break;
      case 13:this.setLocation(p.x - 20,p.y
                                    );break;
      case 14:this.setLocation(p.x + 20,p.y
                                    );break;
      case 15:this.setLocation(p.x - 20,p.y
                                    );break;
      case 16:this.setLocation(p.x + 20,p.y
                                    );break;
      case 17:this.setLocation(p.x - 20,p.y
                                    );break;
      case 18:this.setLocation(p.x + 20,p.y
                                    );break;
      case 19:this.setLocation(p.x - 20,p.y
                                    );break;
      case 20:this.setLocation(p.x + 10,p.y
                                );break;} } }
  protected void stopTimer() {
    t.stop(); }
  public static void main(String[] args) {
    Trema t = new Trema();}
}
```

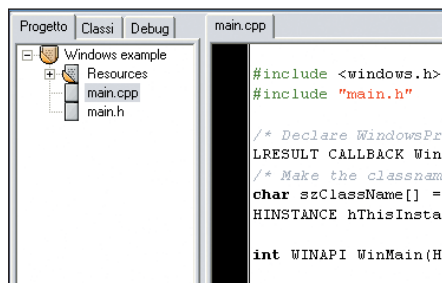
Le API di windows comandate da DEV C++

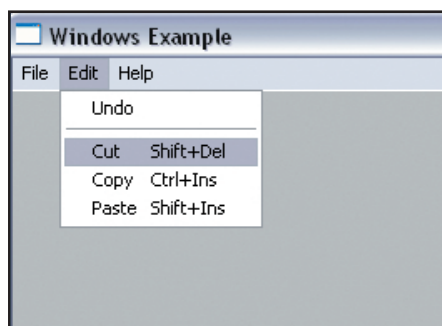
Il linguaggio C++ è noto per la sua estrema versatilità, ne è riprova la sua efficienza nel controllo delle routine primitive di manipolazione di oggetti visuali. Con le Win API è possibile creare, modificare, controllare ed eliminare tutti gli oggetti presenti in windows; quindi finestre standard e dialogo, message box, menu e quant'altro. C++ dalla versione standard, alle differenti versioni proposte dalle maggiori case di software consente un totale controllo delle API di windows. In particolare, la versione con licenza GNU a cui facciamo riferimento, ossia DEV C++ fornisce utili esempi per comprendere al meglio l'utilizzo di tali strumenti. Nell'esperienza proposta verrà aperto e manipolato un progetto di esempio, disponibile con il compilatore citato. Verrà, così, facilmente creata una finestra con un una barra di menu tipica di semplici applicazioni, come il notepad. Ovviamente, le diverse voci associate saranno vuote o richiameranno semplici message box. In altre parole vedremo come sia possibile costruire una struttura di finestra che contenga dei menu, facile da personalizzare e da associare all'applicazione che il programmatore intende sviluppare. L'insieme dei file che si richiameranno fanno parte di un unico progetto. La radice dei programmi (che richiama gli altri) sarà *main.cpp*.

Fabio Grimaldi

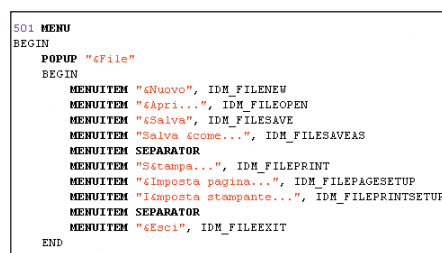
‹1› APRI PROGETTO O FILE ...

Dopo aver lanciato DEV C++ che comprende oltre al compilatore un gradevole IDE (fortemente personalizzabile) e un potente debugger, si aprirà il file di esempio. Per farlo bisogna accedere al menu file e selezionare la voce apri progetto o file. Quindi, selezionare l'esempio che si presta allo scopo.

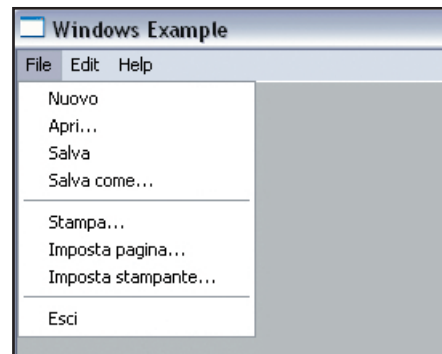
‹2› LA MANIPOLAZIONE DELLE API

L'esempio che si intende testare, ed eventualmente manipolare, si trova sotto la cartella *DEV-Cpp> Exemples> WinMenu*. Per cominciare si può caricare *WinMenu.dev*, in automatico nella parte sinistra dell'ambiente apparirà l'intero progetto, nella tipica struttura ad albero (come esplora risorse, per intenderci). Si possono osservare tutti i componenti del progetto di esempio.

‹3› APERTURA E COMPILAZIONE DEL FILE MAIN.CPP

Dall'insieme di file presenti nel progetto si seleziona *main.cpp*. Successivamente, dal menu *esegui* si sceglie la voce *Compila & Esegui*, come si può notare dall'icona riportata al fianco, la stessa cosa si può ottenere cliccando sull'apposito tasto nella barra degli strumenti. In automatico si produrrà il file eseguibile che viene anche lanciato.

‹4› OSSERVIAMO IL RISULTATO OTTENUTO

Il risultato ottenuto è una finestra con barra di menu. Con le tre voci file, *edit* e *help*. Ognuna contenente sottomenu. L'applicazione sviluppata è una struttura, per cui ad ognuna delle voci presenti non è associato alcuna routine, se non semplici message box. Come passo successivo si può provare a modificare la finestra ottenuta.

‹5› MANIPOLANDO IL FILE RSRC.RC

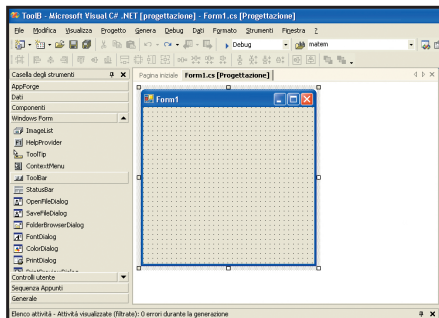
```
501 MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Nuovo", IDM_FILENEW
        MENUITEM "&Apri...", IDM_FILEOPEN
        MENUITEM "&Salva", IDM_FILESAVE
        MENUITEM "Salva &come...", IDM_FILESAVEAS
        MENUITEM SEPARATOR
        MENUITEM "S&tampa...", IDM_FILEPRINT
        MENUITEM "&Imposta pagina...", IDM_FILEPAGESETUP
        MENUITEM "I&mposta stampante...", IDM_FILEPRINTSETUP
        MENUITEM SEPARATOR
        MENUITEM "&Esci", IDM_FILEEXIT
    END
```

Un modo per personalizzare l'applicazione è manipolare il file delle risorse. Come esercizio proviamo a tradurre il menu file in italiano. Dal progetto selezioniamo il file *rsrc.rc* ci posizioniamo nel codice alla funzione associata al menu file e sostituiamo tutte le parole con la traduzione italiana (*open* con *apri, new* con *nuovo* e così via).

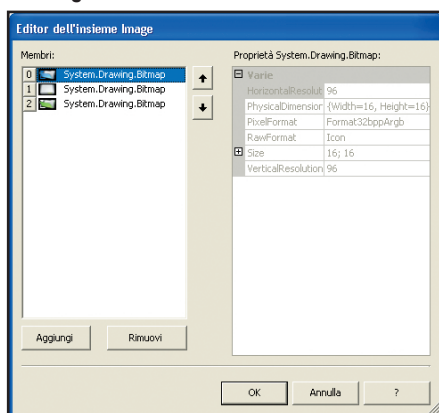
‹6› ECCO IL NUOVO RISULTATO OTTENUTO

Il nuovo risultato ottenuto è sostanzialmente simile al precedente, è la versione italiana. Sono disponibili anche gli accessi accelerati, attraverso la contemporanea pressione del tasto *ALT* con l'iniziale della voce, o comunque con la lettera preceduta dal simbolo *&* (in fase di costruzione del menu) come specificato nel codice al passo precedente.

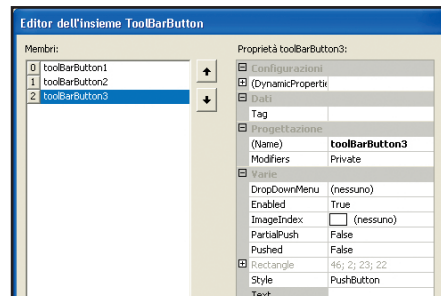
Una ToolBar nelle tue applicazioni Con Visual Studio .NET è un attimo!

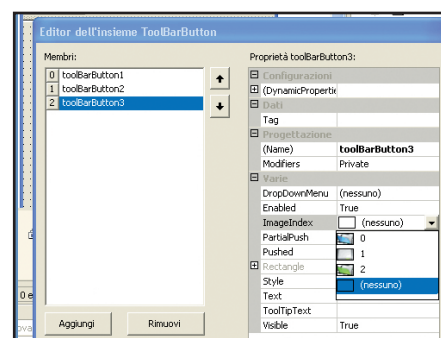
Benché sia presente nella stragrande maggioranza delle applicazioni commerciale, la toolbar è la cenerentola delle applicazioni sviluppate per uso personale. In questa occasione ci ripromettiamo di mostrare quanto sia semplice e “divertente” creare un toolbar con Visual Studio. La maggior parte delle operazioni sono visuali e si riducono a un drag&drop, mentre la parte di programmazione si limita a definire le operazioni da associare alla pressione di vari tasti della toolbar. Vedrete che il lavoro da fare si può dividere in due parti: la definizione della barra vera e propria e l’organizzazione delle immagini da associare ai singoli pulsanti. A questo proposito, ci sono moltissime risorse Web che mettono a disposizione, gratuitamente, immagini per le icone. Date un’occhiata alla directory di Google: *Computers > Software > Desktop Customization > Utilities > Icons.* Seguendo il tutorial, arealizzeremo una barra risulterà dall’aspetto professionale e perfettamente in linea con gli standard Microsoft. Insomma, non avete più scuse: è il momenti sistemare una barra di pulsanti nelle vostre applicazioni.

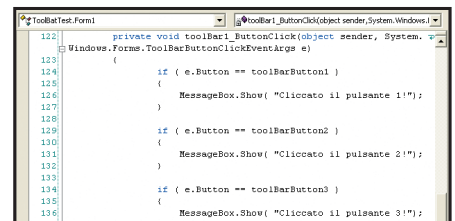
Raffaele del Monaco

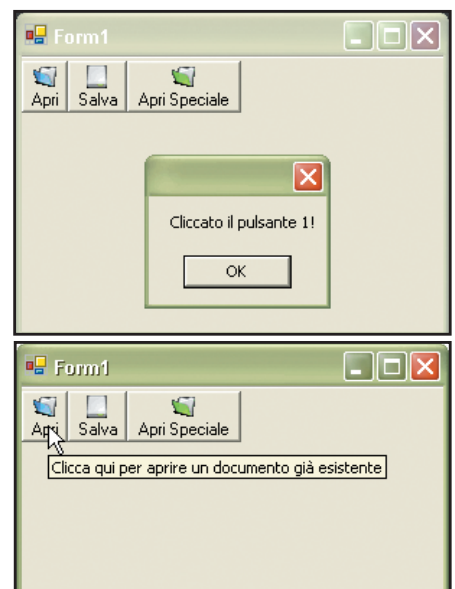
‹1› PREPARIAMO LA FORM PER IL NUOVO PROGETTO

Avviamo Visual Studio e creiamo una nuova applicazione C#, io l’ho battezzata *ToolB*, a voi la scelta del nome che preferite!
Dalla toolbox, trasciniamo una *ToolBar* ed una *ImageList* sulla form.

‹2› SCEGLIAMO LE IMMAGINI TRA QUELLE PRESENTI SU HD

Tra le proprietà della *ImageList*, selezioniamo Images e, nella scheda che appare, andiamo ad aggiungere le immagini che andranno associate ai pulsanti della toolbar.
Il pulsante *Aggiungi* consentirà di indicare la posizione dei relativi file.

‹3› ASSOCIAMO LA TOOLBAR ALLA I,AGELIST

Selezioniamo la *ToolBar* e, nel pannello proprietà, andiamo alla voce *ImageList.* Dal relativo menu a discesa, indichiamo il nome della *ImageList1,* cioè quella che abbiamo appena riempito di immagini. Andiamo alla voce *Buttons* e, con un clic sul pulsante dei tre puntini, si aprirà una maschera che ci consentirà di aggiungere tutti i pulsanti che vogliamo alla nostra *ToolBar.* Per ogni pulsante, basta un clic su *Aggiungi.*

‹4› UN’IMMAGINE PER OGNI PULSANTERI

Sempre restando nell’editor di *ToolBarButton*, per ogni pulsante possiamo selezionare l’immagine associata, scelta dalla *ImageList1*. È sufficiente andare a selezionare l’immagine che ci interessa dal menu a discesa della proprietà *ImageIndex* di ciascun pulsante. La proprietà *Text* permette invece di specificare il testo che accompagna ogni pulsante.

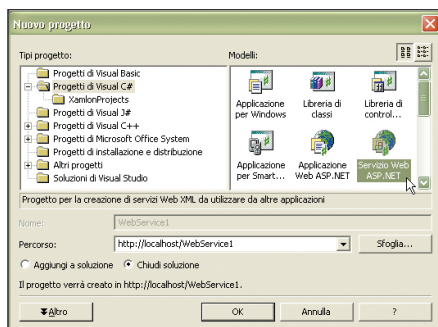
‹5› UN PO’ DI CODICE

Non resta che aggiungere un gestore per l’evento di clic su ogni pulsante. Come al solito, Visual Studio .NET rende banale questa operazione: un doppio clic sulla tool bar e verrà generato il codice-cornice, in cui definire le azioni da compiere per ogni pulsante. Nell’esempio apriamo un box differente per ogni pulsante.

‹6› LA NOSTRA APPLICAZIONE

Ecco il risultato dei nostri sforzi… un po’ spartano, per la verità! In figura, trovate anche una tooltip a fianco del primo pulsante. Per attivarla, è sufficiente selezionare l’omonima voce nelle proprietà di ogni pulsante e indicare il testo che si vuol far apparire.

Impariamo a costruire un Web Service con Visual Studio .NET

La piattaforma .NET, già dal nome, rivela la sua stretta voglia di Internet. È innegabile che la programmazione per il Web ed in particolare per i Web Services abbia ricevuto un notevole impulso dall'affermarsi della piattaforma Microsoft. La semplicità con cui è possibile creare e consumare servizi Web grazie a .NET e a Visual Studio lasciò molti di noi a dir poco impressionati. Eppure, forse proprio per l'estrema semplicità della "messa in opera" di un Servizio Web, se ne discute sempre poco. Lo si dà per scontato. C'è il Wizard, è necessario IIS, non ci si deve preoccupare di nulla... così viene spesso liquidata la questione. Non sulle pagine di ioProgrammo, ovviamente! Torniamo dunque su una argomento che in altre occasioni e con maggiore urgenza abbiamo trattato: prendetelo come un "richiamino" in forma di Express per metter su un Web Service senza perdere la testa.

Raffaele del Monaco

‹1› UN MAGO PER OGNI ESIGENZA

Abbiamo Visual Studio .NET aperto. Possiamo non utilizzare l'immancabile wizard? Come al solito: apriamo un nuovo progetto, indichiamo nome e posizione, e specifichiamo che sarà un servizio Web.

‹2› HELLO WORLD COME SERVIZIO WEB

```
public Service1()
{
    //CODEGEN: chiamata richiesta da Progettazione
                                    servizi Web ASP.NET.
    InitializeComponent();
}
// [WebMethod]
// public string HelloWorld()
// {
//    return "Hello World";
// }
```

Le informazioni relative a nome e posizione, sono sufficienti a Visual Studio per generare il codice di un semplice Web Service, pronto per essere interrogato. Quello riportato sopra è proprio una parte del codice generato: come vedete, è sufficiente eliminare il commento alle righe che definiscono il metodo HelloWorld() per avere subito attivo il metodo. Questo scheletro non é certo funzionale alla produzione di un web service efficace, ma l'esempio evidenzia come Visual Studio.NET sia stato concepito proprio per favorire lo sviluppo di questa tecnica.

‹3› IMPOSTIAMO I METODI

```
[WebMethod]
public string Saluta(String strMyName)
{
   return "Ciao" + strMyName;
}
[WebMethod]
public string ChiSei()
{
   return "Sono il tuo primo WS!";
}
```

Ovviamente a noi non basta avere un metodo che, senza accettare parametri, ci mandi un bel saluto. Modifichiamo allora il codice, così come proposto sopra: due metodi pubblici, di cui il primo accetta un parametro e ritorna un valore. Notate che i metodi che vogliamo siano richiamabili come servici Web, oltre che pubblici, devono essere preceduti dalla dichiarazione *[WebMethod].*
Più facile di così...

‹4› DEFINIAMO NOME E DESCRIZIONE

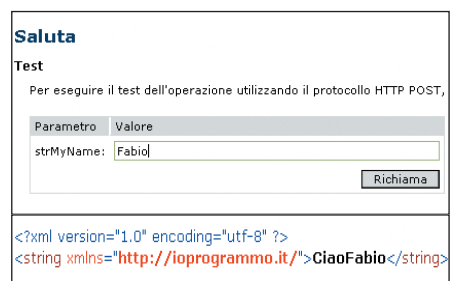
```
[WebService(
    Namespace="http://ioprogrammo.it/",
    Name="Il mio primo Web Service con .NET",
    Description="Un servizio semplice semplice da
                                  testare subito"
)]
```

Beh, non ci crederete, ma il Web Service è pressoché completo e già pronto per funzionare. Manca giusto qualcosa che ne rendano facile la fruizione anche a chi non ne conosce la struttura. Le righe di codice presentate sopra vanno aggiunte subito prima della dichiarazione del servizio web (public class Service1).
Come vedete, abbiamo dichiarato un Namespace, in modo da rendere univoca l'identificazione del servizio, e abbiamo impostato nome e descrizione per rendere più chiara la finalità e la modalità di utilizzo a chi dovrà interrogare il servizio.

‹5› AVVIAMO IL SERVIZIO

Cliccando su F5, Visual Studio compilerà e avvierà il servizio. Contemporaneamente, verrà avviata un'istanza di Explorer che mostrerà la pagina *http://localhost/WebService1/Service1.asmx*, ovverosia l'indirizzo del nostro servizio. Vedete che sono presenti tre link: il primo punta alla file WSDL con la descrizione del servizio, mentre glil altri due consentono di provare i metodi attraverso il browser.

‹6› UN TEST DIRETTO

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://ioprogrammo.it/">CiaoFabio</string>
```

Per provare il servizio è sufficiente cliccare su uno dei due link proposti. Ad esempio, clicchiamo su *Saluta*. La pagina Web che si aprirà consentirà di inserire un nome: in realtà stiamo simulando, via Web, il passaggio di parametri che dovrà avvenire da codice. Inserendo una stringa e cliccando sul pulsante *Richiama*, otterremo il documento XML che rappresenta la risposta del nostro metodo all'invocazione. Tutti i meccanismi di scambio dati SOAP e quant'altro rimangono nascosti al programmatore che non si deve preoccupare di implementare la parte più a basso livello del codice.

Connettersi a un DB in ASP in ambiente Dreamweaver

L'uso dei Database nell'ambito del web è sempre più frequente. Gli ambienti visuali di sviluppo come Front Page e Macromedia sono molto usati per la creazione di semplici pagine html. Accanto alle più frequenti funzioni per la creazione di siti web tali software mettono a disposizione dei programmatori utili tools per la manipolazione di database; ad esempio mediante la metodologia ASP (active server pages). Come tutti i linguaggi ASP può essere programmato con il più spartano degli editor. Innegabili però, sono i vantaggi che si possono avere con l'uso di un ambiente di sviluppo visuale e integrato come Macromedia Dreamweaver. Esso da un lato è ottimo strumento per lo creazione di pagine html, pronte a essere pubblicate sul web, dall'altro si presta alle soluzioni di moltissime situazioni correlate allo sviluppo su piattaforme web. Nel caso specifico integra ottimamente ASP e la tecnologia sottesa all'uso e alla manipolazione di DB da web. Purché abbiate un server web (Pws, Apache o IIS) installato su pc potrete in pochi passi connettere un DB al sito in costruzione. La connessione mediante ODBC è lo stadio preliminare (utilizzando il software in esame). Una volta connesso il DB al server mediante ASP è possibile comandarlo semplicemente programmando. Passo zero per il raggiungimento dell'obbiettivo è aprire il software sopraccitato, si tratta della versione MX 2004.

Fabio Grimaldi

‹1› AVVIO DEL SOFTWARE E CREAZIONE DEL SITO WEB

Dopo aver avviato Macromedia Dreamweaver definiamo un nuovo sito dal omonimo menu.
Selezionando gestisci sito appare una nuova finestra di dialogo dove sono presenti tutti i siti che si stanno sviluppando. La lista è vuota se è la prima volta che utilizziamo il software.
Scegliamo la voce nuovo.

‹2› CREAZIONE GUIDATA DELLE CARATTERISTICHE DEL SITO

La creazione guidata del sito e delle sue caratteristiche è accompagnata da svariati passi associati alla stessa finestra di dialogo.
Dopo aver immesso il nome del sito (chiamato Libri come l'omonimo DB), cliccando sua avanti viene chiesto se si vuole utilizzare la tecnologia server. Possiamo scegliere uno dei vari linguaggi proposti. In questo caso desideriamo l avorare con asp;, perciò selezioniamo ASP javascript.

‹3› SCELTA DELL'AREA DI LAVORO COME CARTELLA LOCALE

Lo sviluppo può essere fatto provando direttamente i file su di un server collegato in remoto al nostro PC.
Noi optiamo per la strada più semplice ed efficiente in fase di primo sviluppo.
Scegliamo la prima voce: *modifiche e prove* localmente su una cartella a nostra scelta. *C:\inetpub\wwwroot* è il default a cui fa riferimento IIS.

‹4› COMUNICAZIONE TRA L'URL E IL SERVER DI PROVA

La comunicazione tra URL e il server di prova avviene attraverso il conosciuto protocollo HTTP. Si rende necessario un indirizzo che per default è *http://localhost/libri*.
Manteniamo tale indirizzo e dopo averlo testato con un apposito click su prova URL chiudiamo la procedura di creazione del sito.

‹5› CONFIGURAZIONE DEL DSN (DATA SOURCE NAME)

Dal menu elabora o direttamente sul segno + dell'apposita finestra applicazioni ASP definiamo il DNS. Con questo procedimento si legherà il DB precedentemente creato in Access (ci si può connettere anche ad altri formati di DB). Dalla voce seleziona scegliere il file mdb che si intende connettere al sito in fase di sviluppo (nel esempio libri.mdb).

‹6› DEFINIZIONE DELLA CONNESSIONE

Nella successiva finestra di dialogo tra i DSN presenti oltre al DB di prova apparirà quello appena connesso. Si nomina la nuova connessione LinkLibri e si termina la fase prefissata. Cliccando sul tasto prova si potrà verificare se l'operazione è avvenuta correttamente. Adesso il DB è pronto ad essere usato nel sito. Tutto é pronto. Continuando a lavorare con DreamWeaver avrete adesso a disposizione un database da potere utilizzare in modo del tutto visuale, ad esempio trascinando una datagrid sulla pagin, automaticamente verrà generato il codice che consente di visualizzare il contenuto del database.

Connettersi a un DB in ASP in ambiente Dreamweaver

L'uso dei Database nell'ambito del web è sempre più frequente. Gli ambienti visuali di sviluppo come Front Page e Macromedia sono molto usati per la creazione di semplici pagine html. Accanto alle più frequenti funzioni per la creazione di siti web tali software mettono a disposizione dei programmatori utili tools per la manipolazione di database; ad esempio mediante la metodologia ASP (active server pages). Come tutti i linguaggi ASP può essere programmato con il più spartano degli editor. Innegabili però, sono i vantaggi che si possono avere con l'uso di un ambiente di sviluppo visuale e integrato come Macromedia Dreamweaver. Esso da un lato è ottimo strumento per lo creazione di pagine html, pronte a essere pubblicate sul web, dall'altro si presta alle soluzioni di moltissime situazioni correlate allo sviluppo su piattaforme web. Nel caso specifico integra ottimamente ASP e la tecnologia sottesa all'uso e alla manipolazione di DB da web. Purché abbiate un server web (Pws, Apache o IIS) installato su pc potrete in pochi passi connettere un DB al sito in costruzione. La connessione mediante ODBC è lo stadio preliminare (utilizzando il software in esame). Una volta connesso il DB al server mediante ASP è possibile comandarlo semplicemente programmando. Passo zero per il raggiungimento dell'obbiettivo è aprire il software sopraccitato, si tratta della versione MX 2004.

Fabio Grimaldi

‹1› AVVIO DEL SOFTWARE E CREAZIONE DEL SITO WEB

Dopo aver avviato Macromedia Dreamweaver definiamo un nuovo sito dal omonimo menu. Selezionando gestisci sito appare una nuova finestra di dialogo dove sono presenti tutti i siti che si stanno sviluppando. La lista è vuota se è la prima volta che utilizziamo il software. Scegliamo la voce nuovo.

‹2› CREAZIONE GUIDATA DELLE CARATTERISTICHE DEL SITO

La creazione guidata del sito e delle sue caratteristiche è accompagnata da svariati passi associati alla stessa finestra di dialogo. Dopo aver immesso il nome del sito (chiamato Libri come l'omonimo DB), cliccando sua avanti viene chiesto se si vuole utilizzare la tecnologia server. Possiamo scegliere uno dei vari linguaggi proposti. In questo caso desideriamo l avorare con asp;, perciò selezioniamo ASP javascript.

‹3› SCELTA DELL'AREA DI LAVORO COME CARTELLA LOCALE

Lo sviluppo può essere fatto provando direttamente i file su di un server collegato in remoto al nostro PC. Noi optiamo per la strada più semplice ed efficiente in fase di primo sviluppo. Scegliamo la prima voce: *modifiche e prove* localmente su una cartella a nostra scelta. *C:\inetpub\wwwroot* è il default a cui fa riferimento IIS.

‹4› COMUNICAZIONE TRA L'URL E IL SERVER DI PROVA

La comunicazione tra URL e il server di prova avviene attraverso il conosciuto protocollo HTTP. Si rende necessario un indirizzo che per default è *http://localhost/libri.* Manteniamo tale indirizzo e dopo averlo testato con un apposito click su prova URL chiudiamo la procedura di creazione del sito.

‹5› CONFIGURAZIONE DEL DSN (DATA SOURCE NAME)

Dal menu elabora o direttamente sul segno + dell'apposita finestra applicazioni ASP definiamo il DNS. Con questo procedimento si legherà il DB precedentemente creato in Access (ci si può connettere anche ad altri formati di DB). Dalla voce seleziona scegliere il file mdb che si intende connettere al sito in fase di sviluppo (nel esempio libri.mdb).

‹6› DEFINIZIONE DELLA CONNESSIONE

Nella successiva finestra di dialogo tra i DSN presenti oltre al DB di prova apparirà quello appena connesso. Si nomina la nuova connessione LinkLibri e si termina la fase prefissata. Cliccando sul tasto prova si potrà verificare se l'operazione è avvenuta correttamente. Adesso il DB è pronto ad essere usato nel sito. Tutto é pronto. Continuando a lavorare con DreamWeaver avrete adesso a disposizione un database da potere utilizzare in modo del tutto visuale, ad esempio trascinando una datagrid sulla pagin, automaticamente verrà generato il codice che consente di visualizare il contenuto del database.

SOFTWARE SUL CD

Apache 1.3.33/2.0.52

Uno dei server Web più usati al mondo

Un indispensabile ormai. ioProgrammo lo ripropone spessissimo fornendovi sempre le versioni più aggiornate. Viene utilizzato in tutti i problemi che riguardano un'integrazione con il Web.

Se avete bisogno di un server Web per provare le vostre applicazioni Web, oppure da usare in sistemi di produzione, Apache è quello che fa per voi.

Directory: /Apache/

PHP 5.0.3

Il linguaggio di scripting per il web

Ormai PHP lo conoscete tutti, e se non lo avete mai usato, sicuramente vi sarà capitato di accedere a qualche sito web sviluppato con PHP. Saprete dunque perciò che è un linguaggio di scripting particolarmente utilizzato per sviluppare Web Application. Incredibilmente potente, fa della completezza del linguaggio, della facilità di apprendimento, della capacità di integrarsi con applicazioni di Dababase i suoi punti di forza.

Directory: /PHP

Python 2.4

Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà

Viene usato in una varietà di applicazioni. A quanto pare è largamente utizzato ad esempio da Google per lo sviluppo delle loro applicazioni. Si caratterizza per la gestione dinamica della memoria, per l'elevata portabilità, per la curva di apprendimento relativamente breve. Un linguaggio di programmazione di cui sentiremo parlare a lungo.

Directory: /Python

Hibernate 3.0

Il framework per gestire la persistenza dei dati

Sicuramente avrete già avuto occasione di notare che ioProgrammo comincia a dare spazio a Hibernate. Si tratta di un Framework molto interessante, di cui sicuramente te parleremo in profondità. Lo scopo principale è garantire una corrispondenza fra il modello ad oggetti dei programmi che sviluppiamo e lo schema dei database sottostanti.

Directory /Hibernate

MySQL 4.1.9

Il server di database OpenSource più diffuso al mondo

MySQL è un indispensabile. ioProgrammo ogni mese vi offre la versione aggiornata. Si tratta del server di database fondamentale per la maggior parte delle applicazioni Internet scritte in PHP. Ma è difussissimo anche per le applicazioni Standalone e grazie ai nuovi connector comincia a essere usato anche dagli sviluppatori .NET

Directory /Mysql

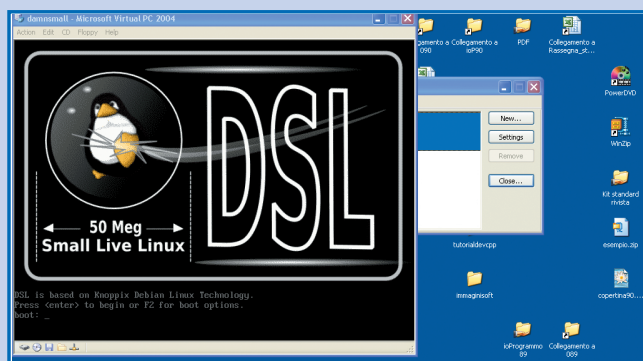
Sqlite 2.8.15

Il nuovo database Bundled con PHP5

Come molti di voi già sapranno, il nuovo PHP5 ha spostato l'attenzione dal supporto a MySQL a quello a SQLite. SQLite è un database piccolo e leggero che consente di

Damn Small Linux

Linux in 50MB

Si tratta di una versione live dell'ormai noto Linux. Per Live si intende che potere masterizzare la traccia ISO che distribuiamo in questo numero di ioProgrammo ed effettuare il boot del sistema operativo direttamente dal CDRom senza dovere scrivere neanche un file sull'Hard Disk. Nonostante questo, le prestazioni di Damn Small sono buone, potete tranquillamente configurare la rete, navigare con FireFox, persino leggere la posta. Nei 50Mb in questione non è stato possibile concentrare un intero compilatore o un ambiente di sviluppo, tuttavia rimane una buona opportunità per i programmatori che desiderano incominciare a prendere confidenza con l'ambiente, magari per sviluppare applicazioni multipiattaforma. Nello screenshot notate come abbiamo fatto girare la Damn Small in una finestra di Windows tramite Virtual PC. Con un sistema abbastanza ben dotato si riescono a tenere in piedi contemporaneamente i due sistemi senza grande difficoltà.

Directory :/damnsma

lavorare su archivi di dati pur non dovendo installare un server. Per molti versi lo si può paragonare ad Access ma le sue caratteristiche lo rendono particolarmente versatile e compatibile con la maggior parte delle web application scritte in PHP. Il software che vi presentiamo è un piccolo client a linea di comando, utile per gestire, creare tabelle e database senza dover passare per PHP.
Directory: /Sqlite

SharpDevelop 1.0.3.1761

L'alternativa a Visual Studio a basso costo

Microsoft senza dubbio ha modificato il suo modo di intendere lo sviluppo delle applicazioni con l'introsuzione della piattaforma .NET. Tuttavia lo sviluppo in tecnologia .NET non è una prerogativa dei soli ambienti targati Microsoft.
Esistono delle pur valide alternative a Visual Studio se si vuole comunque sviluppare in C# ad esempio oppure in VB.NET, SharpDevelop è una di queste. Visual Studio rimane sempre un ambiente estremamente completo colmo di caratteristiche che lo rendono unico, tuttavia SharpDevelop rappresenta un'ottima alternativa, economica, potente, affidabile.
Directory: /Sharpdevelop

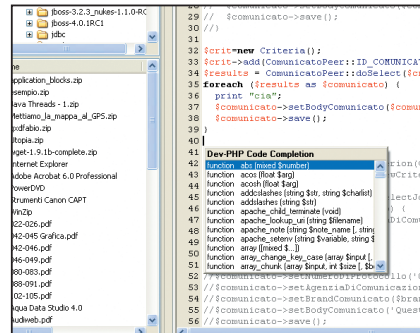
Tomcat 5.5.4

Il servlet container per Java e JSP

L'idea è molto semplice. Sviluppare in Java pagine Web. Ad un primo sguardo, Tomcat, potrebbe sembrare un normale WebServer. Ed in effetti è un normale WebServer! In grado di soddisfare le richieste per qualunque pagina Html. In realtà però Tomcat è anche qualcosa in più. Di fatto però Tomcat offre qualcosa in più, ovvero la capacità di soddisfare richieste per applicazioni Java. Potrebbe sembrare complesso, in realtà lo è meno di quanto sembri. Immaginate Tomcat come un grande contenitore al cui interno ci sono altri contenitori ciascuno dei quali rappresenta un'applicazione Java, che richiamata da luogo ad una pagina HTML interpretabile da un browser. Questo consente di sviluppare pagine Web (JSP) utilizzando tutta la potenza della normale gerarchia di classi Java e la sintassi e il linguaggio che qualunque programmatore Java conosce bene.
Directory: /tomcat

Dev-PHP 2.0.9

Ottimo editor PHP OpenSource

Se state iniziando a sviluppare in PHP avrete bisogno di un editor. Scrivere codice con il notepad può essere un esercizio divertente, ma quando iniziate a scrivere script leggermente più complessi si impone la scelta di passare a un editor più completo.

DEV-PHP non solo è completo ma anche molto potente. Dotato di code completion, sintax highlighting, funzionalità di ricerca avanzate ed una serie di tool piuttosto interessanti rappresenta una grande scelta per programmare in PHP. Inoltre è un editor straordinariamente leggero, oltre che gratuito ed OpenSource.
Da non perdere!
Directory: /DevPHP

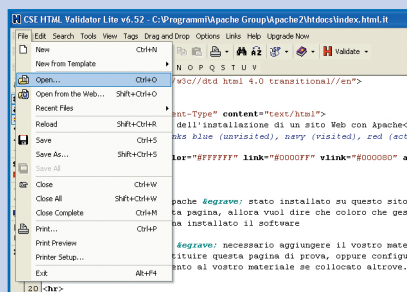
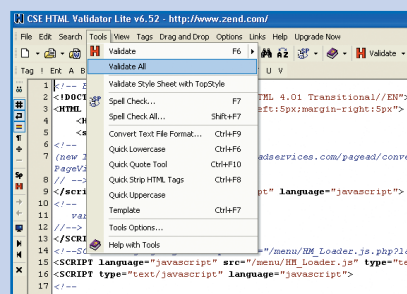
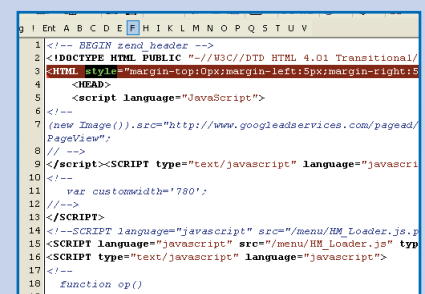
Irrlicht 0.7

Sviluppare Giochi 3D potenti in modo semplice

Da quando Alfredo Marroccelli ha iniziato la serie di articoli riguardanti lo sviluppo di Giochi 3D basati sul 3D Engine Irrlicht, siamo stati letteralmente sommersi dalle email dei lettori che si sono divertiti a programmare il proprio gioco 3D. Irrlicht è straordinariamente potente, facile da usare, intuitivo. Tanto da meritare uno spazio fisso nelle nostre pagine.
Directory: /irrlicht

Cse HTML Validator Lite V6.52

Un validatore per le vostre pagine html

Il Web non è più un mondo completamente selvaggio come ai suoi inizi. È importante produrre delle pagine HTML conformi agli standard, accessibili e visualizzabili da tutti i sistemi se si vuole ottenere un qualche risultato dal proprio lavoro. Cse Validator è un validatore HTML. Controlla che il codice prodotto all'interno delle vostre pagine sie conforme agli standard dettati dal W3C consortium e vi avverte se avete usato qualche tag in maniera non corretta o se qualche link non è funzionante.
Si tratta di un ottimo tool da utilizzare in fase di finalizzazione dei propri lavori. Produrre dei siti accessibili e conformi agli standard è decisamente un dovere per un buon programmatore, oltre che un ottimo biglietto di presentazione ai vostri clienti.
Directory /csevalidator

1 APRIAMO UN NUOVO FILE - Dal menu *File/Open* selezioniamo il file html su cui eseguire la validazione

2 VALIDAMO IL FILE - Dal menu *tools* scegliamo *validate all* per ottenere la validazione

3 I RISULTATI - Nella finestra in basso verranno evidenziati tutti i problemi di validazione

Smarty 2.6.7

Il template engine per PHP

Se avete letto il libro “imparare PHP” allegato alla versione Plus di ioProgrammo nello scorso numero, sapete già cosa è Smarty e come si usa. Si tratta di un template engine per PHP.
Ovvero un sistema che consente di separare la logica di programmazione da quella di layout.
Sostanzialmente il web designer avrà a doisposizione un formato molto simile all’HTML per sviluppare le pagine che conterranno l’elaborazione delle applicazioni sviluppate con PHP. Molto comodo e potente è certamente un sistema da utilizzare per sviluppare in modo efficace le proprie Web Application con PHP

Directory /Smarty

Sqlite .NET

Un connector .NET per questo interessante database

Sqlite è il nuovo database bundled con la versione 5 di PHP. Al di là di godere di questa interessante caratteristica, Sqlite è anche un ottimo database che può essere utilizzato in applicazioni standalone. Gode della proprietà di essere molto leggero e file based, non necessità di particolare attenzione per essere installato e con poche righe di codice il vostro DB sarà pronto. Il meccanismo è molto simile a quello di Access, il sistema di autenticazione e gestione degli utenti è infatti demandato al sistema operativo, tuttavia la velocità e l’affidabilità sono impressionanti. Se avete bisogno di un database leggero senza troppe complicazioni ma non volete affidarvi ad Access, Sqlite è quello che fa per voi. Questo connector per .NET vi consente di accedere a database formato Sqlite dalle vostre applicazioni .NET.

Directory: /Sqlite

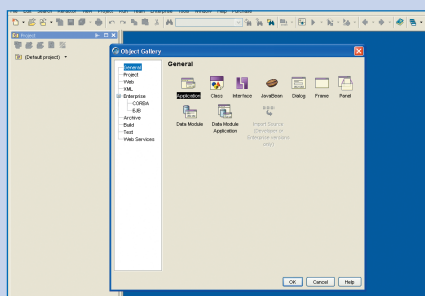
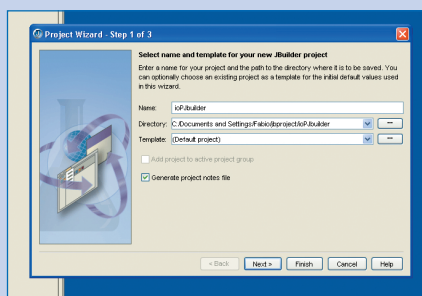
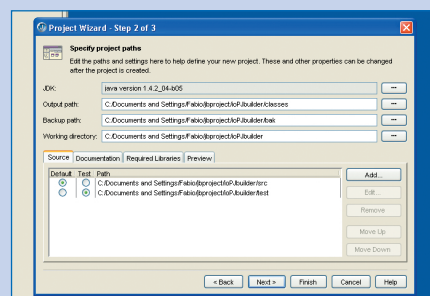
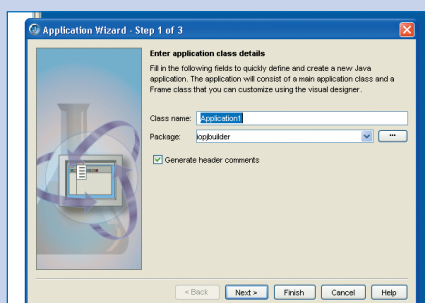
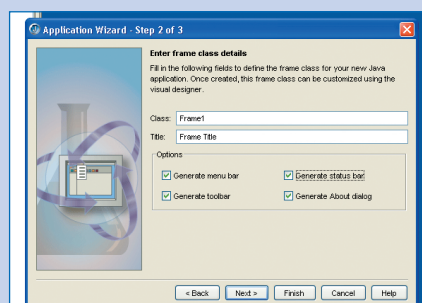
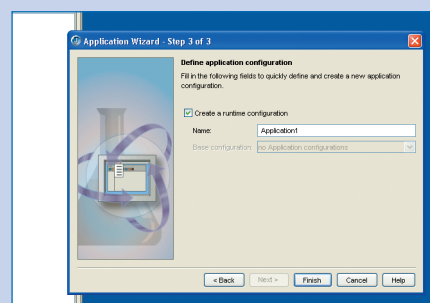
Aqua Data Studio 4.0

Il query builder leggero ed efficiente

Aqua Data Studio è un buon tool per la gestione di Dabase di diverso tipo. Supporto

BORLAND JBUILDER 2005 FOUNDATION

L’eccezionale ambiente di Borland per lo sviluppo Java

Jbuilder è l’ambiente proposto da Borland per lo sviluppo di applicazioni Java. Come è tradizione di Borland si tratta di un ambiente incredibilmente curato nei particolari. Dotato di tutte le caratteristiche essenziali per un ottimo editor Java, la versione Foundation ha dalla sua parte il costo praticamente nullo, dovuto alla sua licenza, l’uso della foundation è infatti gratuito per usi personali. Il software è completo e senza scadenza. Certo ha delle limitazioni rispetto alla versione enterprise, ad esempio non è possibile gestire i progetti UML, ma si tratta di limitazioni che comunque influiscono solo sulla produzione di progetti di largo respiro. Insomma per un programmatore Java che si accinge a iniziare oppure per un programmatore Java che non ha necessità di sviluppare progetti Enterprise, la versione Foundation è assolutamente da provare. Molto comodo il Visual Designer

1 UNA NUOVA APPLICAZIONE - Si parte del menu *File/New/* scegliendo poi l'icona *Application* nella finestra di dialogo successiva

2 DIAMO UN NOME AL PROGETTO - Scegliamo il nome da dare al progetto indicandolo nell'apposita casella. Clicchiamo poi su *ok*

3 QUALE JAVA? - Scegliamo la versione di java da utilizzare, con JBuilder viene distribuita la versione 1.4

5 DIAMO UN NOME ALL'APLICAZIONE - Scegliamo un nome per l'applicazione, può differire da quello del progetto generale

6 LO SCHELETRO - Scegliamo di generare un'applicazione completa di menu, icone e status bar

7 CREAIMO EVENTUALI FILE DI CONFIGURAZIONE - Se l'applicazione necessiterà di file di configurazioni particolari è possibile generarli da qui

Oracle, MySQL, MS SQl e moltissimi altri formati. Non è completamente visuale come qualcuno potrebbe aspettarsi da un prodotto pubblicizzato come query builder ma è dotato di alcune interessanti funzionalità che lo rendono uno strumento di supporto rapido ed efficiente. Ad esempio nonostante le vostre query debbano essere descritte manualmente, Aqua Studio è dotato di completion e syntax hilghlighting il che lo configura come un editor per il linguaggio SQL piuttosto che come un Rad Query builder. Se utilizzato correttamente può velocizzare di molto lo sviluppo di applicazioni di database. Naturalmente può essere utilizzato anche come rimpiazzo dei vari PHPMyAdmin o MySqlCC anche se non trattandosi di un prodotto dedicato non è dotato di tutte le funzionalità tipiche di questi strumenti.

Directory /Acquadatastudio

Eclipse 3.0.1

La piattaforma universale multifunzione

Sembra un sottotitolo esagerato ed eclatante: "La piattaforma universale multifunzione" ed invece Eclipse è nato proprio con questo scopo. A prima vista sembra un normale editor per programmatori Java, anzi molto di più che un normale editor, visto che ospita una serie di funzionalità piuttosto avanzate che vanno dal code completion alla syntax highlithing al refactoring e che lo stanno portando a diventare uno standard proprio per Java, tuttavia è anche vero che Eclipse è completamente estensibile per mezzo di plugin, tanto che può essere utilizzato da programmatori PHP come da programmatori C++ e persino come frontend verso database etc. Insomma qualunque tipo di necessità voi abbiate, Eclipse è in grado di aiutarvi. Se poi siete dei programmatori Java rientra in quel ristretto numero di software indispensabili per gestire rapidamente il vostro lavoro.

Directory: /Eclipse

Blender 2.36

Un modeller per oggetti 3D

Blender non è propriamente un tool riservato ai soli programmatori, piuttosto può aiutare alcune categorie di programmatori nello sviluppo dei propri software.

Si tratta infatti di un ambiente per lo sviluppo di modelli tridimensionali. Indispensabile per chi per esempio sviluppa videogame gode di alcune importanti proprietà. È un prodotto GPL perciò privo di costi diretti, è straordinariamente potente e completo di tutte le caratteristiche dei più noti software commerciali, è in grado di esportare i dati in formati comprensibili dalla maggior parte degli engine 3D ivi compresi Irrlicht e Directx. Perciò se siete degli sviluppatori di videogame o se in un qualche modo sviluppate dei prodotti attinenti al mondo tidimensionale, Blender senza dubbio vi può essere utile.

Directory: /Blender3D

Menalto Gallery 2.0

Una gallery fotografica pronta per il Web

Siete appassionati di fotografia? Volete vendere online sfondi, immagini? Più semplicemente volete creare una galleria fotografica Online? Questo software è quello che fa per voi. Si tratta di una delle prime web application ad essere nate per la gestione di gallerie fotografiche sul web, perciò vanta un'esperienza notevole e tutto questo si riscontra nella cura dei particolari. È un software scritto in PHP che si appoggia in parte a MySQL in parte utilizza imagemagick oppure le GD per la gestione delle immagini. Sufficientemente semplice da installare, è facilmente integrabile in moltissimi CMS opensource, che anzi spesso e volentieri lo propongono come tool aggiuntivo. In ogni caso rappresenta un'ottima soluzione per la gestione di gallerie di immagini sul Web

Directory: /gallery

HSqldb 1.7.3

Un database piuttosto potente

Nato in sordina come prodotto OpenSource, HSqlDB in breve tempo sta conquistando gli onori della cronaca grazie alla sua eccezionale velocità, semplicità d'uso e affidabilità. Si tratta di un Database completamente scritto in Java che occupa appena 100k ma che espone caratteristiche interessanti. Sembrerebbe che proprio HSqlDb sia stato scelto come prodotto di base per lo sviluppo del nuovo applicativo di database che completerà la suite di OpenOffice nella sua versione 2.0. Questo testimonia anche la validità del prodotto, se ce ne fosse ulteriore bisogno.

Directory /hsqldb

TikiWiki

Un Wiki piuttosto interessante

Probabilmente il miglior Wiki opensource attualmente disponibile. Scritto in PHP con l'ausilio dei template di Smarty si sta lentamente avviando ad essere un Wiki esteso, cioè un wiki che integra funzionalità tipiche di un Cms. Ma cosa è un wiki? In linea del tutto generale un wiki è un

---

che consente di disegnare le form per trascinamento dei componenti, come nella migliore tradizione degli ambienti RAD, tuttavia richiede un sistema sufficientemente dotato per potere essere utilizzato con profitto.

Directory /JbuilderFoundation

4 CARATTERISTICHE GENERALI - Settiamo le proprietà generali, per i nostri scopi andranno bene quelle di default

8 PRONTI - Lo scheletro della nostra prima applicazione è pronto cliccate su Design per completare l'interfaccia

sistema che consente di scrivere tramite un editor html online delle pagine web molto semplici, la particolarità sta nel fatto che se in una pagina è contenuta una parola del tipo "DocumentazioneStorica" per esempio, cioè spezzata con le lettere maiuscole, verrà automaticamente creata una nuova pagina avente come titolo le due parole ricongiunte, il link verrà posizionato nella pagina originale e sarà possibile editare la seconda pagina con un sistema ad albero, inoltre ogni qualvolta in una pagina verrà inserita la parola originale, automaticamente verrà creato un link. La seconda particolarità di un wiki è che normalmente viene utilizzato come forma di scrittura collaborativa, ovvero gli utenti autorizzati possono apportare delle modifiche alle pagine, creando una sorta di revisione. Per tutti questi motivi i Wiki sono molto utilizzati per la creazione rapida di documentazione.
Directory /tikiwiki

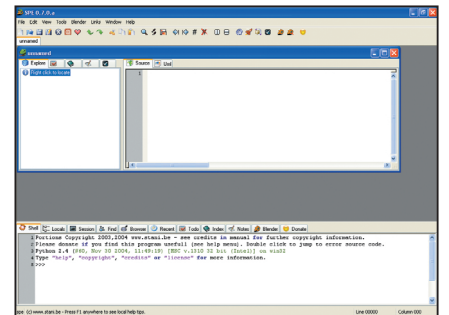
Zope 2.7
L'application server basato su Python

Molti di voi saranno abituati a sviluppare pagine Web utilizzando PHP, ASP.NET e alcuni JSP. Altri addirittura utilizzeranno Perl, eppure è possibile sviluppare per il web utilizzando Python. Questo tipo di sviluppo comporta una serie di vantaggi come quello di avere a disposizione un linguaggio multipiattaforma, object oriented, molto elegante e flessibile, inoltrre python si sta sempre di più affermando come linguaggio di scripting base per la gestione dei sistemi operativo. Linux in quasi tutte le distrubuzioni ha almeno uno script di gestione basato su Python, ma molte distribuzioni ne fanno addirittura l'asse portante. Anche in ambiente Windows Python si sta diffondendo rapidamente, così scegliere di adottare questo linguaggio anche per lo sviluppo di applicazioni Web può rappresentare una scelta di continuità.
Directory: /Zope

MySQL Connector
Per utilizzare MySQL direttamente da applicazioni .NET, Java, Python

MySQL sta rapidamente diventando uno standard. Se fino a ieri lo standard per le applicazioni sviluppate in ambienti Microsoft era MS SQL Server, rapidamente a questo prodotto si sta affiancando proprio MySQL. Allo stesso modo MySQL sta diventando uno standard per Java e persino per Python. Tutto quello di cui avete bisogno per utilizzare MySQL in modo nativo direttamente da uno di questi ambienti è un "Connector" che fornisca al linguaggio il layer per la connessione e la gestione. In questo numero trovate tutti i connector di cui avete bisogno.
Directory: /Connector

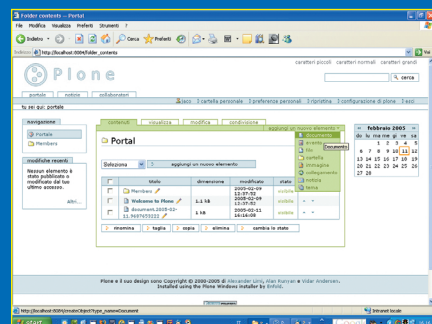
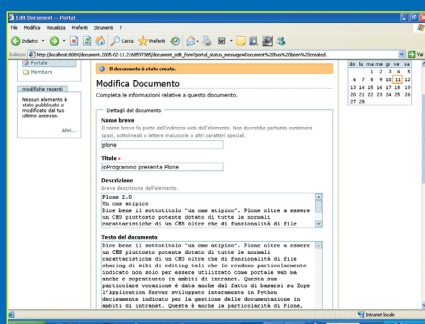
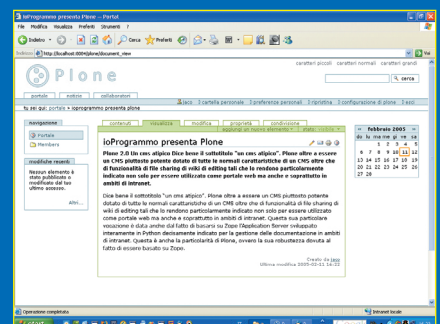
Lua
Per dotare i propri programmi di un ambiente di scripting

Non è inusuale che i software commerciali siano dotati della possibilità di essere estesi tramite un linguaggio di scripting proprietario.

Tipicamente per dotare i propri software di questa funzione, è necessario programmare un intero motore di scripting, in alternativa si può integrarvi all'interno un engine già pronto come ad esempio LUA. Ci parla diffusamente di questa tecnica Alfredo Marroccelli nell'articolo proposto in questo stesso numero di ioProgrammo.

PLONE 2.0
Un cms atipico

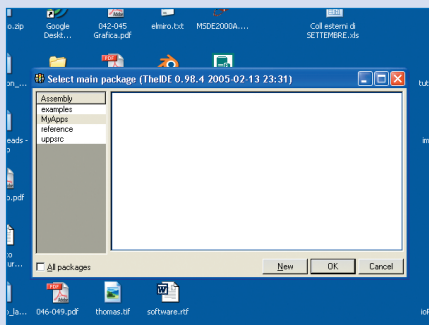
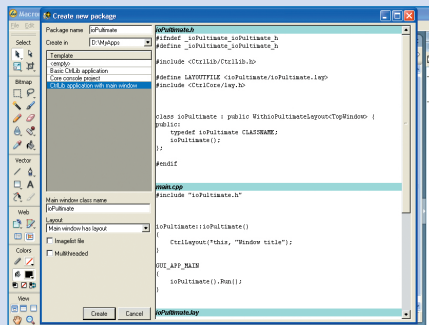
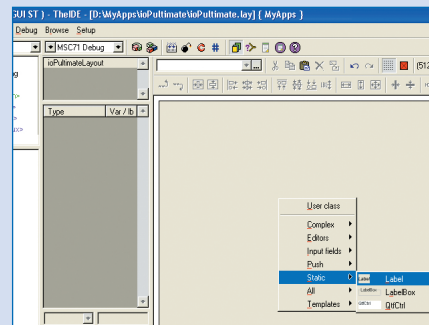
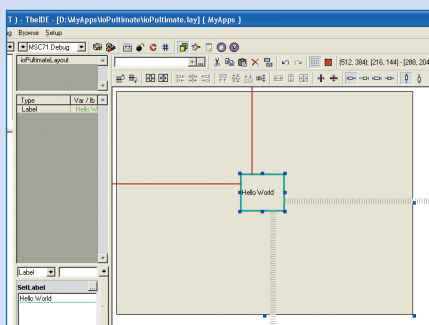
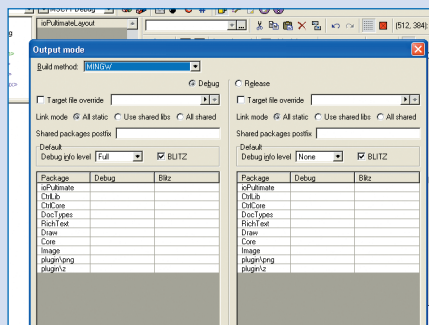
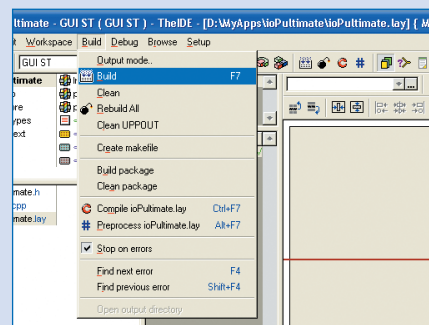
Dice bene il sottotitolo "un cms atipico". Plone oltre a essere un CMS piuttosto potente dotato di tutte le normali carattaristiche di un CMS oltre che di funzionalità di file sharing di wiki di editing tali che lo rendono particolarmente indicato non solo per essere utilizzato come portale web ma anche e soprattutto in ambiti di intranet. Questa sua particolare vocazione è data anche dal fatto di basarsi su Zope l'Application Server sviluppato interamente in Python decisamente indicato per la gestione delle documentazione in ambiti di intranet. Questa è anche la particolarità di Plone, ovvero la sua robustezza dovuta al fatto di essere basato su Zope.

Directory /Plone

PUBBLICARE CON PLONE

1 SCEGLIAMO IL TIPO DI DOCUMENTO - Fra quelli disponibili individuiamo quale tipo di pubblicazione desideriamo effettuare

2 COMPILIAMO I DATI - Nella pagina successiva inseriamo tutti i dati necessari per la pubblicazione

3 PUBBLICATO - Il documento verrà pubblicato online con modalità che dipendono dai ruoli di chi ha inserito il documento

ULTIMATE++

Un fantastico ide "quasi RAD" per le applicazioni C++

Si tratta di un ambiente di sviluppo per applicazioni C++, nella versione allegata a ioProgrammo. Lo trovate abbinato al compilatore Mingw; nonostante questo può essere utilizzato anche con altri compilatori. L'ambiente offre tutte le caratteristiche classiche di un ambiente professionale, parliamo di code complexion, debugging, sintax highlighting. La caratteristica più interessante dell'IDE è che è dotato di un minimo di funzionalità RAD il che dato i bassi costi del pacchetto lo rende particolarmente attraente agli occhi degli sviluppatori.

1 CREIAMO UN NUOVO PROGETTO - Dopo averlo lanciato Ultimate++ ci chiede cosa vogliamo fare. Nel mostro caso sceglieremo MyApps e poi new per dire che vogliamo creare un'applicazione personalizzata.

2 IMPOSTIAMO LE CARATTERISTICHE - Scegliamo il nome dell'applicazione e la sua posizione sull'Hard Disk, ma soprattutto indichianmo che vogliamo che la nostra applicazione sia dotata di un layout.

3 DISEGNAMO L'INTERFACCIA - Selezioniamo il componente di tipo "lay" nella colonna di sinistra. Verrà mostrata la form che compone l'applicazione. Cliccando con il tasto destro del mouse inseriamo una label.

4 IMPOSTIAMO LA LABEL - Nella colonna centrale scegliamo "Set label" e digitiamo l'etichetta che vogliamo venga mostrata sulla form. Aiutiamoci con il mouse per posizionare la label.

5 SCEGLIAMO IL COMPILATORE - Clicchiamo sulla finestrelle che indica il compilatore nella barra del menu per scegliere a quale strumento vogliamo affidare la compilazione dell'applicazione.

6 COMPILIAMO - Usiamo il tasto F7 per lanciare la compilazione oppure scegliamo "Build" dall'omonimo menu. L'eseguibile sarà disponibile in una directory accessibile come "Output directory".

All'interno della directory oltre ai file necessari per eseguire la tecnica trovate anche LuaIde, ovvero un comodo IDE proprio per la creazione di script basati su LUA.

Directory: /Lua

Sphinx4

Il motore di riconoscimento vocale

In questo numero di ioProgrammo trovate un articolo completo su come costruire un'applicazione a comando vocale. Il motore vocale che fa girare questo tipo di applicazione è proprio Sphinx4. Interamente scritto in Java rappresenta un motore ad elevate prestazioni, riesce a riconoscere un numero elevato di parole anche in un discorso senza interruzioni.

Leggete l'articolo e scoprirete immediatamente tutte le potenzialità di Sphinx4

Directory: /Sphinx4

PHP-GTK

Creare GUI con PHP

Chi lo dice che PHP è solo un linguaggio di scripting per il Web? Sì certo è usato prevalentemente per creare script lato server tesi a fare funzionare delle web application. È anche vero che è possibile utilizzare PHP per creare potenti script anche lato server. PHP-GTK è un layer che consente di utilizzare i widget di GTK+ per costruire delle GUI piuttosto potenti da utilizzare in script completamente Standalone.

Directory: /PHP-GTK

MySQL Administrator

Il front-end per amministrare mysql in modo grafico

La grande diffusione di MySql degli ultimi tempi è avvenuta anche grazie al proliferare di interfacce grafiche che aiutano l'utente nella sua amministrazione ordinaria. È il caso di MySQL Administrator che fornisce all'utente un potente, completo quanto comodo sistema per la gestione di ogni aspetto di mySQL. Si parte

dalla creazione degli utenti fino alla gestione dei permessi, alla creazione dei database, alle statistiche sull'uso del server.
Directory: /mysql-administrator

MyODBC
Il connector universale per MySQL

Se volete sviluppare un'applicazione che faccia uso di MySQL ma non disponete di un connector nativo, è gioco-forza fare uso di MyODBC. Vero è che ormai esistono connector per qualsiasi tipo di linguaggio. Quasi tutto quello che serve lo trovate allegato a questo numero di ioProgrammo, altrettanto vero è che ci sono dei casi in cui è ancora opportuno utilizzare MyODBC. Il caso più tipico è quello delle pagine ASP, se disponete di un sito scritto in ASP e volete farlo interagire con MySQL non potete fare a meno di utilizzare MyODBC.
Directory: /MyODBC

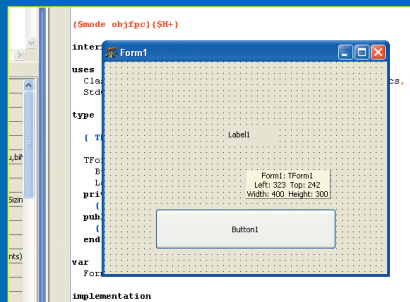
MySQL Query Browser
L'editor SQL per MySQL

Anche in questo caso si tratta di un editor visuale. Attenzione non RAD ma semplicemente visuale. Le query vengono costruite in parte in modo manuale in parte è possibile utilizzare dei selettori sotto forma di bottoni che aiutano a sviluppare query sintaticamente corrette. MySQL Query Browser è un tool che può aiutare in tutte quelle situazioni dove è necessario costruire query complesse che devono essere salvate, riviste, rieditate più di una volta prima di arrivare a un risultato soddisfacente.
Directory: /mysql-query-browser

LAZARUS 0.9.4
Un ambiente RAD più compilatore Freeware per object Pascal

Volevate imparare a programmare in Delphi ma non avevate la possibilità di comperare il costoso ambiente di casa Borland? Lazarus è ciò che fa per voi. Si tratta di un clone Freeware del noto Delphi. La somiglianza è incredibile! Il modo di procedere altrettanto! Lazarus è un RAD funziona con la stessa logica di Delphi, supporta la VCL, è dotato di tutti i componenti classici di Delphi, è sufficientemente veloce e affidabile. Certo, non è il Borland Delphi 2005 con tutte le sue infinite possibilità, ma rappresenta un'ottima base per chi vuole sviluppare applicazioni Standalone utilizzando la produttività classica di un ambiente RAD come quelli famosi di casa Borland. Si tratta di un tool eccezionale che non mancherete di iniziare ad apprezzare per tutte le sue caratteristiche.

Directory /Lazarus

1 IL PRIMO PASSO - Inizialmente lazarus si presenta in maniera molto simile a Delphi. Con la barra dei componenti, la palette delle proprietà a sinistra e la form al centro che aspetta di essere riempita.

2 POSIZIONARE I COMPONENTI - Trasciniamo sulla form una componente di tipo label e uno di tipo button, possiamo scegliere di ridimensionarli e posizionarli utilizzando le maniglie laterali e il mouse.

3 AGGIUNGERE IL CODICE - Clicchiamo due volte sul bottone per editare l'evento onClick è aggiungiamo il codice come in figura. Avviamo l'applicazione usando il pulsantino verde in alto a destra.

Spe 0.7.0
Un editor evoluto per Pyhton

La popolarità di un linguaggio si evince anche dal numero di tool e di librerie che vengono prodotte a suo supporto. Se dal punto di viste delle librerie Python può vantare un conspicuo numero di prodotti che ne estendono le caratteristiche, dal punto di vista invece dei tool ancora non avevamo visto degli editor sufficientemente evoluti da poter supportare pienamente lo sviluppo con Python. Spe colma questa lacuna. Si tratta di un editor decisamente evoluto completo di code completion e di sintax highlighting oltre che di tutte le caratteristiche tipiche di un editor professionale. Sia che stiate imparando a programmare in Python sia che siate dei programmatori abituali, sicuramente Spe rappresenta un'ottima scelta.
Directory: /Spe

Tortoise CVS
Ottimo tool per il controllo della revisione

Se lavorate in progetti collaborativi o semplicemente vi interessa tenere traccia delle modifiche che fate nel tempo al vostro codice, sicuramente saprete cosa è un CVS. Tortoise è uno dei frontend più utilizzati per la connessione, gestione e utilizzo di progetti che fanno uso di un server CVS per la tracciabilità delle revisioni.
Directory: /tortoisecvs

Mantaray
Un completo sistema distribuito per la messaggistica aziendale

Anche in questo caso allegato alla rivista trovate un articolo che mostra in dettaglio come utilizzare Mantaray. In particolare Mantaray sfrutta JMS, Java Messaging Service per la gestione e lo smistamento di messaggi.
La particolarità risiede nel fatto che il tool prevede un'architettura completamente distribuita.
mantaray_1.2.1_bin.zip

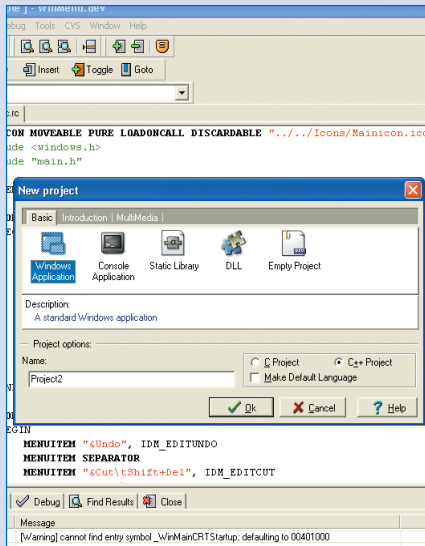
Phpmq
Integra JMS in applicazioni PHP

PHPMQ sfrutta l'architettura di Mantaray per estendere il meccanismo di Java Messaging Service anche ad applicazioni PHP. Se avrete la pazienza di leggere l'articolo all'interno della rivista. Scoprirete come questa possibilità possa ri-

DEV C++

Un editor C++ a basso costo

Dev C++ è un editor distribuito su licenza GPL, come tale non ha costi relativi al diritto d'autore. Come tutto o quasi tutto il software GPL la sua economicità non è affatto sinonimo di scarsa qualità. Al contrartio Dev C++ è uno degli editori più amati ed utilizzati da chi sviluppa in C++. Le caratteristiche sono notevoli. Si va dal Debugger integrato, al project Manager, al Class Browser, al Code Completion. L'insieme di queste caratteristiche, oltre una leggerezza innata dell'ambiente lo rende particolarmente comodo da utilizzare per sviluppare progetti C++ anche di grandi dimensioni

Directory: /Devc++

1 CREIAMO UN NUOVO PROGETTO - Scegliamo "*Windows Application*" e assegnamo un nome al progetto. Dev-C++ creerà per noi lo scheletro dell'aplicazione

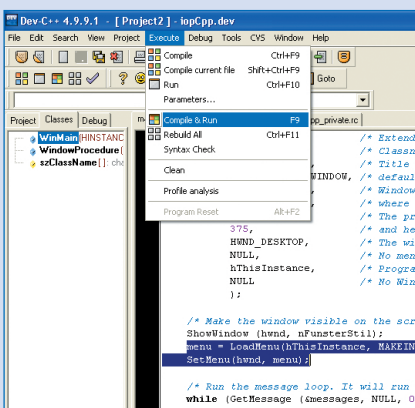
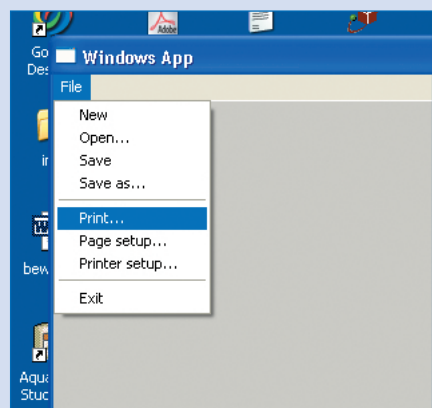
```
#include <windows.h>
#include "main.h"
501 MENU
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "&New", IDM_FILENEW
    MENUITEM "&Open...", IDM_FILEOPEN
    MENUITEM "&Save", IDM_FILESAVE
    MENUITEM "Save &as...", IDM_FILESAVEAS
    MENUITEM SEPARATOR
    MENUITEM "&Print...", IDM_FILEPRINT
    MENUITEM "Page se&tup...",
                    IDM_FILEPAGESETUP
    MENUITEM "P&rinter setup...",
                    IDM_FILEPRINTSETUP
    MENUITEM SEPARATOR
    MENUITEM "E&xit", IDM_FILEEXIT
  END
END
```

2 CREIAMO UN FILE DI RISORSE - Scegliere il menu *File/New/ Resource file*. Salveremo il tutto con il nome *menu.rc*. In questo file abbiamo definito l'aspetto grafico del menu.

```
#define ID_MENU 501
#define IDM_FILENEW          200
#define IDM_FILEOPEN         201
#define IDM_FILESAVE         202
#define IDM_FILESAVEAS       203
#define IDM_FILEPRINT        204
#define IDM_FILEPAGESETUP    205
#define IDM_FILEPRINTSETUP   206
#define IDM_FILEEXIT         207
```

3 DEFIAMO GLI ID - A ogni voce di menu deve essere associata una costante numerica. Eseguiremo questa associazione in un file *main.h*

```
#include <windows.h>
#include "main.h"
[...]
WNDCLASSEX wincl; /* Data structure for
                     the windowclass */
HMENU menu;
[...]
menu = LoadMenu(hThisInstance,
           MAKEINTRESOURCE(ID_MENU));
SetMenu(hwnd, menu);
```

4 MODIFICHIAMO LO SCHELETRO - È arrivato il momento di richiamare il menu all'interno dell'applicazione e associarlo a una finestra, è esattemente quello che facciamo qui

5 AVVIAMO IL TUTTO - Dal menu Execute scegliamo "*compile ed execute*" per dare il via alla compilazione e all'esecuzione del progetto

6 OSSERVIAMO I RISULTATI - Se tutto è andato a buon fine otterremo un risultato come quello in figura, con il nostro nuovo menu pronto per essere utilizzato

velarsi piuttosto interessante nella creazione di intranet aziendali.

mantaray_1.2.1_bin.zip

SevenZip 4.10

Uno dei programmi con migliore algoritmo di compressione per file

Dotato ovviamente di sorgenti, Seven-zip ha davvero un altissimo rapporto di compressione. Il fatto che i sorgenti siano disponibili e il programma basato su LPGL lo rende particolarmente interessante e adatto a essere studiato.

Non di meno utilizzare Seven ZIP porta sicuramente dei vantaggi. Supporta moltissimi tipi di file fra cui: 7z, ZIP, CAB, RAR, ARJ, GZIP, BZIP2, TAR, CPIO, RPM e DEB.

Si integra perfettamente con la shell di Windows, è localizzato in 46 linguaggi ed ha una versione a linea di comando che lo rende adatto a essere usato per creare dei file batch.

Directory: /sevenzip

Analisi di algoritmi di scheduling

Il Kernel ai raggi X

Il kernel è lo strato più vicino alla macchina, si occupa di compiti di cruciale importanza nella gestione del SO. Impariamo come il tempo viene diviso fra le varie applicazioni

Utilizza questo spazio per le tue annotazioni

Conosciamo gli scopi di un SO (sistema operativo) e i principali compiti che assolve. Ci siamo, nello scorso appuntamento, soffermati sul ruolo e funzionamento dei processi. Abbiamo capito come questi ultimi ricoprano la parte da protagonista nella gestione della CPU, e come tale gestione, che un SO è chiamato ad assolvere, sia di notevole rilevanza per sistemi multiprogrammati. Si è osservato come "concedere" a più processi che ne fanno richiesta, il più importante hardware, che non a caso è indicato come il "cuore del computer", ossia la CPU, sia una mansione niente affatto banale. Il compito diventa ancora più arduo se come siamo abituati a fare dobbiamo rispettare parametri di "buon" funzionamento dell'intero sistema. Si sono così ideati molti algoritmi che prevedono la concessione della CPU (risorsa non divisibile ma comunque interrompibile) rispettando diversi parametri di ottimizzazione. Si tratta degli algoritmi di scheduling. I due fondamentali algoritmi di scheduling il *FCFS (first come first served)* e *SJF (shortest job first)* sono stati esaminati per primi. Si tratta di due metodi event driven, in cui l'assegnazione della CPU avviene in funzione del verificarsi di alcuni eventi e il processo deterrà la preziosa risorsa CPU fin quando non avrà terminato il suo lavoro. Nel panorama, a dire il vero molto ricco, delineato da tutti gli algoritmi rimane da studiare un altro del tipo event driven, a priorità; un algoritmo che segue la logica time driven, il round robin, e infine un metodo misto a coda multilivello. Per il primo dei metodi indicati appronteremo un'analisi dettagliata seguita da simulazione attraverso la produzione di un programma C++.
I restanti saranno accennati e forse sviscerati in futuro. Infine, verrà descritta una comparazione tra i metodi.

ALGORITMO A PRIORITÀ

Il metodo FCFS nello scegliere tra i processi disposti nella coda di *ready,* (pronti a essere eseguiti *-run*) segue una logica *FIFO,* quindi dalla lista dei processi estrae il più "anziano", ossia quello che da più tempo attende. *SJF* invece, sceglie il processo che ha una minore richiesta di tempo; in tal modo si garantisce un buon tempo turnaround medio (ricircolo), ossia mediamente i vari processi dovranno attendere meno rispetto al caso precedente. Il limite di *SJF* è l'impossibilità di conoscere il tempo richiesto di CPU dai singoli processi, che può essere solo stimato, con prevedibili errori a volte rilevanti. Una soluzione sempre più usata fa uso di priorità. Ad ogni processo viene assegnata una priorità, ossia un numero compreso in un intervallo. Range usati sono *[0, 7]* e *[0, 4095]*. Tra i processi pronti ad essere eseguiti si sceglie quello a priorità maggiore, che ha il numero più grande. Alcuni algoritmi considerano come processi ad alta priorità quelli che hanno numeri vicini allo zero. Questo è un sistema molto più flessibile dei precedenti, poiché la priorità può essere espressa in funzione di molti parametri. Ad esempio, se si produce il livello di priorità in funzione della sola stima del tempo richiesto da un singolo processo, allora, evidentemente il metodo "degenera" nel *SJF.* Ma

CPU E MEMORIA

La CPU e la memoria rappresentano le più importanti risorse di cui dispone un elaboratore. In ambiente multiprogrammato hanno però caratteristiche differenti. Mentre la prima non è divisibile, la memoria può essere partizionata in più parti da assegnare contemporaneamente a diversi processi che ne fanno richiesta. La CPU, però, si può interrompere nella sua attività e quindi essere "concessa" a più processi solo in diversi tempi. La contemporaneità di esecuzione di più task è pertanto solo un'impressione che deriva dalla concessione molto veloce a più processi per fissati intervalli di tempo.

nel caso più generale la prelazione si può ottenere in funzione di molti fattori, che possono essere stabiliti in parte anche dall'utente che produce il processo il quale può fissare un valore di partenza. In generale i fattori che contribuiscono sono: le richieste di risorse, il comportamento del processo in esecuzione, il tipo di processo (qualora il SO sia in grado di effettuare una classificazione) e comunque eventi esterni. Ad ogni modo, non è molto rilevante il come si produca la priorità, a mio avviso è utile sapere che esiste questo utile strumento che può essere anche configurato dal sistemista. Prima di passare alla simulazione del sistema osserviamo dei possibili rischi di cui potrebbe soffrire il metodo. Ad esempio, processi a bassa priorità, nel caso in cui le code di *ready* siano regolarmente "affollate", potrebbero rischiare di essere eseguiti molto "tardi" non garantendo tempi di risposta minimi spesso imposti come specifiche di sistema (tipici dei SO real time). Per il problema specifico si è trovata una soluzione che permette di servire il processo in questione aggiungendo alle componenti che contribuiscono alla determinazione della priorità anche l'anzianità conosciuta in gergo come *aging priority.*

IL CODICE

Il programma riportato di seguito vuole essere una semplice simulazione di un algoritmo a priorità. Per rendere il tutto più semplice e comprensibile alcuni eventi che nella realtà avvengono in seguito ad *interrupt,* solitamente gestiti mediante programmazione concorrente, nel caso specifico sono simulati mediante input manuali da console. Anche le strutture dati sono semplificate per focalizzare l'attenzione sulle dinamiche che muovono le scelte dell'algoritmo di priorità, per cui anziché trattare un PCB nella sua completezza avremo a che fare con una struttura che contiene le informazioni salienti di un processo, identica cosa dicasi per le altre strutture come i tracciati dei log. Eventuali altre scelte saranno esposte in fase di commento del codice prodotto. Il programma proposto è nella sua versione integrale. La prima parte riporta ovviamente la struttura dati espressa come una serie di variabili semplici e strutturate entrambe globali, visibili quindi all'intero ambito del programma.

```
// simulp.cpp
// Semplice programma per la simulazione del
// metodo priorità per l'assegnazione della CPU
// ai processi in stato di ready.
// valutazione di alcuni parametri.
#include<iostream.h>
#include<string.h>
// numero massimo di processi
const int nmaxproc=100;
// numero massimo di azioni (ass. CPU)
const int nmaxlog=200;
// stima tempo medio di sistema tra una ass. e la succ.
const int tsys=1;
// tempo attuale
int t=0;
// numero corrente di log
int nlog=-1;
// informazioni salienti circa un singolo processo
// in stato di coda di ready
struct tprocesso
{     char nome[8];
      int num, prior, tp;
} processo;
// coda di processi in ready
tprocesso coda[nmaxproc];
// situazione sommativa del sistema
// numero di processi, tempo utente totale utilizzato
// tempo sistema totale utilizzato, tempo inattività
struct tsistema
{     int np, tu, ts, ti;
} sistema;
// traccia delle attività di CPU
struct tlog
{     tprocesso proc;
      int tiniz, tfine;
} logs[nmaxlog];
```

Le prime due costanti descrivono le dimensioni di due array molto importanti; *nmaproc* e *nmaxlog* corrispondono ai vettori *coda* e *logs.* I due array tengono traccia rispettivamente: dei processi che sono in coda di *ready* che verranno opportunamente ordinati in base alla loro priorità e a tutte le attività della CPU per fissate unità di tempo. Esaminando i tipi base che costituiscono tali vettori si comprende meglio il ruolo che ricoprono. *tprocesso* è un tipo struttura (un record) che contiene le informazioni di un singolo processo, dal nome al suo numero d'ordine, e anche la priorità e il tempo richiesto di CPU. Serve una precisazione che proietti la realtà al tipo di simulazione che ho approntato. La priorità sarà un valore numerico compreso nell'intervallo *[0, 4095]* inserito manualmente, ma che, come nella realtà può essere modificato (infatti, la priorità di un processo può variare nel tempo di vita del processo stesso). Il tempo di CPU richiesto va visto come l'effettiva tempo di utilizzo della risorsa e non come una stima. Il fatto che verrà inserito in fase di produzione dell'array coda risponde soltanto ad un'esigenza di comodità. La struct sistema contiene informazioni generali sull'andamento della CPU: il numero di processi in attesa e i vari tempi di utilizzo. I tempi sono: utente come somma degli usi dei vari processi; sistema come tempo impiegato dal SO tra una assegnazione e l'altra di CPU per l'elaborazione della scelta (per semplicità si è considerato fisso, di una sola unità di tempo). Un'unità assume il valore

EVOLUZIONE DI MLS

MLQ con feedback è una evoluzione del sistema a code multiple. Qui un job non è fisso in una coda ma può spostarsi in funzione delle attività generali di sistema da una coda ad un'altra.

che l'utente trova più opportuno assegnarli, può essere facilmente traslata rispetto al microprocessore in uso; può valere un decimo di micro secondo come un nano secondo. Le variabili globali usate sono *t* che indica il tempo corrente, *nlog* la quantità attuale di log, ovvero il numero della generica (corrente) attività della CPU.
Cominciamo con l'esaminare le funzioni di servizio per la configurazione delle variabili e per l'output.

```
// funzione di inizializzazione
void setup()
{    int i;
     t=0;
     nlog=0;
     sistema.tu=0;
     sistema.ti=0;
     sistema.ts=0;
     cout<<"n. processi iniziali -> ";
     cin>>sistema.np;
     for (i=0; i<sistema.np; i++)
     {   coda[i].num=i;
         cout<<"nome processo -> ";
         cin>>coda[i].nome;
         cout<<"priorità (0 - min , 4095 max) -> ";
         cin>>coda[i].prior;
         cout<<"tempo richiesto -> ";
         cin>>coda[i].tp;
     }
};
// visualizzazione dello stato generale del sistema
void visualsys()
{    cout<<"\n situazione attuale di sistema \n"
       <<" n. processi attivi : "<<sistema.np
       <<"\n tempo utente totale utilizzato :"<<sistema.tu
       <<"\n tempo sistema totale utilizzato
                                   :"<<sistema.ts<<'\n';
};
// visualizzazione della coda di ready
void visualcoda()
{    int i;
     cout<<"\n situazione attuale della coda di ready \n";
     for (i=0; i<sistema.np; i++)
     {   cout<<"\n "<<i<<" - processo(
                                 "<<coda[i].num<<"): "
         <<coda[i].nome<<",  priorità: "<<coda[i].prior
         <<",  tempo richiesto: "<<coda[i].tp;
     } };
// Output del log
void visuallog()
{    int i;
     cout<<"\n Storia dell'attività di CPU \n";
     cout<<"nome"<<"\t\t"
            <<"n."<<'\t'
            <<"tempo"<<'\t'
            <<"prior"<<'\t'
            <<"t.iniz"<<'\t'
            <<"t.fine"<<'\n';
     for (i=0; i<=nlog; i++)
     {   cout<<'\n'<<logs[i].proc.nome<<"\t\t"
            <<logs[i].proc.num<<'\t'
            <<logs[i].proc.tp<<'\t'
            <<logs[i].proc.prior<<'\t'
            <<logs[i].tiniz<<'\t'
            <<logs[i].tfine<<'\n';
     }
};
```

Con *setup* vengono inizializzate tutte le variabili prima descritte. Inoltre, vengono inseriti i primi processi. Naturalmente è possibile aggiungerne in itinere, durante la normale attività della CPU. La famiglia delle funzioni visual produce l'output dei dati sensibili quali il vettore *logs* e il vettore coda contenente le informazioni sui vari processi. Durante l'attività di sistema si possono aggiungere nuovi processi che quindi arricchiscono la coda di ready; alcuni di essi possono variare delle informazioni come ad esempio la priorità. Le due attività sono svolte dalle due routine riportate di seguito *aggiunta()* e *aggiorna()*. Contestualmente viene aggiornato il record sistema.

```
// eventuale aggiunta di nuovi processi
void aggiunta()
{   int j,npagg;
    cout<<"\n n. processi da aggiungere (0 - esci) -> ";
    cin>>npagg;
    for (j=0; j<npagg; j++)
    {    coda[j+sistema.np].num=j+sistema.np;
         cout<<"nome processo -> ";
         cin>>coda[j+sistema.np].nome;
         cout<<"priorità (0 - min , 4095 max) -> ";
         cin>>coda[j+sistema.np].prior;
         cout<<"tempo richiesto -> ";
         cin>>coda[j+sistema.np].tp; };
    sistema.np+=npagg; }
// aggiorna la richista di tempo e/o priorità
// dei processi in stato di ready
void aggiorna()
{   int npagg,com;
    visualcoda();
    cout<<"\n n. processo da aggiornare (-1 esci) -> ";
    cin>>npagg;
    while (npagg!=-1)
    {    cout<<"priorità: "<<coda[npagg].prior
           <<"\n nuova (-1 lascia invariata) -> ";
         cin>>com;
         if (com != -1) coda[npagg].prior=com;
         cout<<"tempo da aggiungere (attuale:
                                   "<<coda[npagg].tp
           <<") al processo -> ";
         cin>>com;
         coda[npagg].tp+=com;
         cout<<"\n numero di processo da aggiornare (
                                            -1 esci)";
         cin>>npagg; }  }
```

COMPILATORE USATO

Il compilatore usato è Dev-CPP di bloodshed nella versione 4.2. Un interessante compilatore free dalle potenzialità davvero sorprendenti.

Per potere estrarre il processo a maggiore priorità si può agire o facendo una ricerca esaustiva, quindi sequenziale, sul vettore coda dei processi, oppure ordinando questo ultimo (ovviamente sulla base del campo *prior*) ed estrarre l'elemento in cima. Scegliamo la seconda opzione.

```
void sort()
{    int i,j;
     tprocesso temp;
     for (i=0; i<sistema.np-1; i++)
          for(j=0; j<sistema.np-i-1; j++)
               if (coda[j].prior<coda[j+1].prior)
               {    temp=coda[j];
                    coda[j]=coda[j+1];
                    coda[j+1]=temp; }
};
```

la funzione cruciale assegna la CPU al processo a più alta priorità, dopodiché aggiorna opportunamente i log e le altre variabili globali. Il codice prodotto è abbastanza autoesplicativo, diamogli un'occhiata.

```
// Assegnazione del processo allo stato di run
// inoltre aggiorna le strutture coda, sistema e logs
// nonchè la variabile t
void assegna()
{    int i;
     // attività di sistema
     sistema.ts+=tsys;
     ++nlog;
     strcpy(logs[nlog].proc.nome,"sistema");
     logs[nlog].proc.tp=tsys;
     logs[nlog].proc.num=-1;
     logs[nlog].proc.prior=-1;
     logs[nlog].tiniz=t;
     logs[nlog].tfine=t+tsys;
     t+=tsys;
     // si preleva in cima alla lista
     // processo a maggiore priorità
     ++nlog;
     strcpy(logs[nlog].proc.nome,coda[0].nome);
     logs[nlog].proc.tp=coda[0].tp;
     logs[nlog].proc.num=coda[0].num;
     logs[nlog].proc.prior=coda[0].prior;
     logs[nlog].tiniz=t;
     logs[nlog].tfine=t+coda[0].tp;
     t+=coda[0].tp;
     sistema.tu+=coda[0].tp;
     // traslazione della restante parte di coda
     for (i=1; i<sistema.np; i++)
     {  coda[i-1]=coda[i]; };
     // aggiornamento generale
     sistema.np--;
};
```

Un possibile main program è riportato di seguito. Il ciclo continua fin quando vi sono processi in coda.

Si potrebbe iterare ulteriormente il processo e prevedere tempi di inattività. Estendere il codice a tale eventualità può risultare un piacevole esercizio.

```
// programma principale
// semplice simulazione
int main()
{    setup();
     while (sistema.np!=0)
     { aggiunta();
       sort();
       aggiorna();
       assegna();
       visualsys();
       visualcoda(); };
     visuallog();
     char quit;
     cin>>quit;
}
```

In **Figura 1** si può osservare in che termini si produce l'output. Sono visibili i vari processi, nonché la parola sistema che indica un utilizzo della CPU da parte del SO. Di seguito sono riportati il numero d'ordine del processo, la sua priorità il tempo richiesto e gli intervalli di uso della CPU; insomma un log minimo per tenere traccia delle attività di sistema. Inoltre, sono visibili i tempi totali di utilizzo del sistema. Tali dati possono essere usati per il calcolo di parametri come throughput e cpu activity.

Fig. 1: **Output del programma simulp.cpp**

LOGICA TIME DRIVEN E MISTA

In contrapposizione con una filosofia che tende ad "accontentare" le richieste dei singoli processi per intero in termini di tempo di utilizzo della CPU, si contrappone un metodo che non garantisce ciò. Nei metodi time driver, di cui il round robin ne è la più comune espressione, si parte dal presupposto che ogni processo non può detenere la CPU per un tempo maggiore di un fissato time slice (fetta di tempo). Qualora il time slice non sia sufficiente per soddisfare le richieste di un processo, questi dovrà essere spezzettato in fette di tempo pari al time slice ed essere servito in differenti tempi. Un sistema così organizzato si dice che sia in time sharing.
Ne parleremo nei prossimi numeri.

Fabio Grimaldi